UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (If applicable) 32	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

1. TITLE (Include Security Classification)
Microcomputer Simulation of a Fourier Approach to Optical Wave Propagation

2. PERSONAL AUTHOR(S)
UPTON, John G.

3a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year,Month,Day)	15. PAGE COUNT
Master's Thesis	FROM _____ TO _____	June 1992	114

6. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

7. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Fourier optics, spatial impulse response,diffraction, transient waves, optical propagation

9. ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis uses spatial impulse response theory adapted from continuous-wave Fourier diffraction theory as the basis for a microcomputer program to model pulsed optical wave propagation. Programs to generate uniform circular and uniform square excitation functions are included, along with examples of the spatial impulse response for each. Additionally, two new excitation functions with circular Gaussian and circular Bessel spatial distributions are modelled for use in future research. All programs have been written using the MATLAB software package. This effort provides a means to analyze the transient optical wave propagation of a spatially filtered optical source.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL POWERS, John P.	22b. TELEPHONE (Include Area Code) (408) 646 - 2679	22c. OFFICE SYMBOL EC/Po

DD Form 1473, JUN 86 Previous editions are obsolete.
S/N 0102-LF-014-6603

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Approved for public release; distribution is unlimited.

MICROCOMPUTER SIMULATION OF A FOURIER APPROACH TO OPTICAL WAVE PROPAGATION

by

John G. Upton
Lieutenant Colonel, United States Marine Corps
B.S., United States Naval Academy, 1972

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

ABSTRACT

This thesis uses spatial impulse response theory adapted from continuous-wave Fourier diffraction theory as the basis for a microcomputer program to model transient optical wave propagation. Programs to generate uniform circular and uniform square excitation functions are included, along with examples of the spatial impulse response for each. Additionally, two new excitation functions with circular Gaussian and circular Bessel spatial distributions are modelled for use in future research. All programs have been written using the MATLAB software package. This effort provides a means to analyze the transient optical wave propagation of a spatially filtered optical source.

c.1

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Lasers provide a readily available source of coherent optical radiation for today's technologies. The output of most lasers exhibit a spatial amplitude distribution which is Gaussian. It is possible to spatially filter such a beam to produce an alternatively shaped beam. Such a variation may exhibit a circular or square uniform cross-section and either of these could have an arbitrary spatial weighting distribution. The utility of such filtering is unknown unless the diffracted field distribution can be predicted at any given distance.

Classical scalar diffraction theory provides a means for approximating such a solution. Scalar theory has been shown experimentally to yield accurate results when the diffracting aperture is large compared to the wavelength and the field locations are not too close to the aperture. Classical diffraction analysis is a computationally cumbersome method, however, due to the required solution of multi-dimensional field integrals. For continuous waves, an alternative method [Ref. 1] expresses the solution using the theory of linear time-invariant systems. If the multi-dimensional Fourier transform of the complex field distribution across any plane is taken, the spatial Fourier components can be identified as

plane waves travelling in different directions. The field amplitude at any other point is the sum of each of these contributing waves, if the plane wave's phase shift during travel is accounted for. Thus, the propagation phenomenon may be regarded as a linear space-invariant system characterized by a specific transfer function. This transfer function behaves as a linear dispersive filter with a finite spatial bandwidth. Development of devices which generate ultra-short optical pulses has encouraged the extension of this "Fourier optics" concept to transient waves.

Previous efforts in applying a transfer function approach to transient scalar wave propagation have focused primarily on acoustic applications [Refs. 2,3]. The study of optical diffraction effects utilizing this mathematical approach proves to be a natural extension to that of acoustics. Indeed, the ultrasonic solution is found to be a subset of the optical solution.

A computationally efficient method to model such ultrasonic propagation has been developed by Guyomar and Powers [Refs. 2,3]. Relying upon linear systems theory and using Fourier transforms to transition between the space and spatial transform domains, the solution methodology is an application of the general theory discussed above. It is, as well, quite suitable for efficient computer implementation since it uses Fourier transforms. The mathematical derivation for the optics case in lossless, homogeneous material was

developed by Guyomar [Ref. 4] and is presented in detail in the following chapter.

Computer models of transient acoustic wave propagation have previously been written for both mainframe and microcomputer applications. An example of the latter by Merrill [Ref. 5] was an early effort to investigate the feasibility of generating solutions on microcomputers within a "realistic" length of time (defined arbitrarily and, reasonably, to be thirty minutes). An initial attempt was made using a commercially available program named MATLAB. This is an array-oriented program containing built-in functions, such as the Fast Fourier Transform (FFT) and Bessel function calculations. The capability to incorporate user-developed functions and subroutines also exists. (Additional details of MATLAB will be presented in the third chapter of this paper.) However, because MATLAB memory utilization was limited to 640K at that time, Merrill was forced to abandon the program and to use Microsoft Fortran Version 3.31 to develop his model. The desired maximum run time of thirty minutes was achieved. [Ref. 5]

Advances in the capabilities of microcomputers, including increased speed and enlarged RAM capacity, justified a follow-on effort to develop the appropriate code in MATLAB to be run on upgraded processors, such as the Intel 386/486 series. Such a program would provide a convenient means of researching optical diffraction field characteristics for any excitation

waveform. The specific goal of this thesis was thus defined--to produce a MATLAB program modelling transient scalar optical wave propagation. As no similar published results are known to the author, test and verification of the output results were accomplished by an informal comparison with available unpublished data for the circular- and square-shaped spatial excitation waveforms [Ref. 4]. In addition to these two excitation functions, programs to generate a spatially circular waveform with a selection of either a Gaussian or Bessel amplitude distribution were to be developed for use in further research.

The following chapter describes the wave diffraction problem including the geometry involved, a discussion of the application of linear systems theory, and the mathematical derivation of the field solution utilizing the Fourier approach. Chapter III commences with an overview of the MATLAB language with particular attention paid to the peculiarities in this application. A functional explanation of the program follows within the same chapter. (Appendices C and D contain a detailed explanation of the code.) Chapter IV illustrates the final program outputs for one set of defining parameters. Chapter V summarizes the effort and discusses areas for future research. Throughout these chapters, many figures are utilized to enhance the textual descriptions. These figures can be found at the end of each section within the appropriate chapter of the thesis.

Appendices contain further figures, copies of the elements of the source code, and, as previously mentioned, a detailed explanation of the code's formulation and operation. Lastly, the code is included for each of the four excitation functions.

II. PROBLEM DESCRIPTION

The general solution and explanation detailed here is derived from that presented by Guyomar and Powers in Refs. 2 and 3. They, in turn, point out that the approach is based on the spatial impulse response introduced by Stepanishen [Refs. 6,7,8,9] to describe acoustic propagation and reviewed by Harris [Ref. 10]. All of these references were concerned with acoustic propagation, but the optical field also may be expressed as a temporal convolution of the time excitation with the spatial impulse response. Guyomar and Powers' view differs from Stepanishen's work in that linear systems theory is used to point out the importance of the total impulse response (and its equivalence to the Green's function). Also, the expressions for the spatial impulse response functions are found in the spatial transform domain. In this domain, propagation of the wave will be seen to be the application of two time-varying spatial filters to the spatial spectrum of the input wave.

A. GEOMETRY

The problem utilizes the geometry illustrated in Fig. 1. Given a separable source of excitation in the z=0 plane,

$$\phi_o(x,y,0,t) = s(x,y)\,T(t) \tag{1}$$

the irradiance, or strength of the optical field, at the observation plane located a distance z from the source is to be determined. Note that the excitation is separable in space and time; it is not necessarily separable in x and y. Propagation through a lossless, linear, and homogeneous medium is assumed.

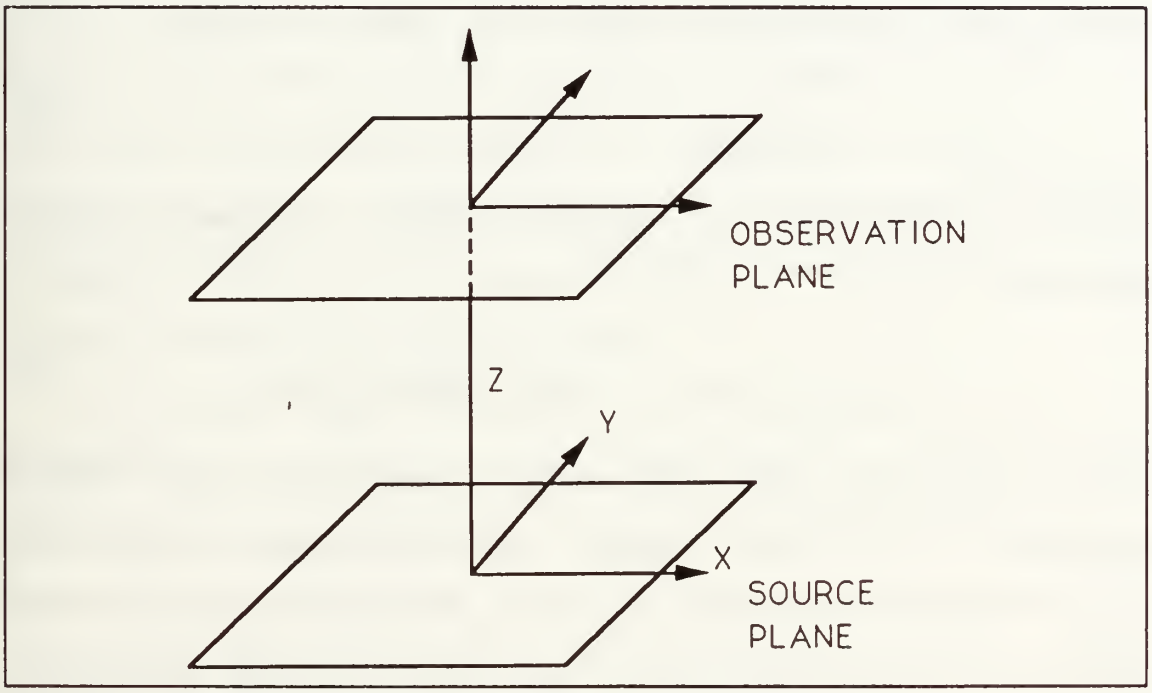

Figure 1. Source and observation planes.

B. LINEAR SYSTEMS SOLUTION APPROACH

It will be shown in the next section [Ref. 4] that the field strength for this separable source can be expressed by

$$\phi(x,y,z,t) = -s(x,y)\,T(t) \underset{x}{*}\underset{y}{*}\underset{t}{*} \frac{\partial g(x,y,z,t)}{\partial z}. \qquad (2)$$

Here, the * indicates convolution over the variable appearing directly below it and g(x,y,z,t) is the impulse response, or Green's function, that both solves the wave equation and satisfies the appropriate boundary conditions. Figure 2 shows the relationships involved.

Factoring out T(t) and its associated temporal convolution, ϕ(x,y,z,t) is also given [Refs. 2,3] as

$$\phi(x,y,z,t) = T(t) \underset{t}{*} h(x,y,z,t) \qquad (3)$$

where h(x,y,z,t) is the "spatial impulse response" and is equal to

$$h(x,y,z,t) = -s(x,y) \underset{x}{*}\underset{y}{*} \frac{\partial g(x,y,z,t)}{\partial z} \qquad (4)$$

where g(x,y,z,t) is the Green's function (or impulse response). In a linear system, the relationship between the spatial impulse response h(x,y,z,t) and the Green's function g(x,y,z,t) is illustrated by the block diagram of Fig. 3. Using the two-dimensional spatial Fourier transform of Equation 4, and

$$\tilde{h}(f_x,f_y,z,t) = \tilde{s}(f_x,f_y)\,F\left\{\frac{\partial g}{\partial z}\right\}, \qquad (5)$$

ϕ(x,y,z,t) can be written as

$$\phi(x,y,z,t) = T(t) \overset{*}{}_{t} F^{-1}\left\{-\tilde{s}\frac{\partial \tilde{g}}{\partial z}\right\} \tag{6}$$

where the tilde notation indicates the transform of the spatial function.

In the spatial frequency domain, the transform of the spatial impulse response is

$$\tilde{h}(f_x, f_y, z, t) = \left\{-\tilde{s}\frac{\partial \tilde{g}}{\partial z}\right\} . \tag{7}$$

The two-dimensional spatial convolutions found in Equations 2 and 4 do not lend themselves readily to computer simulation, due to the integrations involved. However, as suggested by Equations 6 and 7, this difficulty can be overcome by taking the appropriate two-dimensional spatial transforms using Fast Fourier Transform (FFT) techniques.

Viewing Equation 7 once more, $\tilde{s}$ is the angular spectrum of the amplitude distribution of the input function; $\partial\tilde{g}/\partial z$ may be thought of as the "propagation transfer function" related to the medium of propagation. We will find that this propagation transfer function behaves as a time-varying spatial filter that increasingly attenuates the higher spatial frequencies of the source as time increases [Refs. 2,3].

Figure 4 illustrates the concepts discussed thus far in the time domain. To summarize, the general solution technique begins with the wave equation for lossless media and requires determining the Green's function (or the two dimensional

spatial transform of the Green's function) which solves the wave equation and satisfies the boundary conditions.

The inverse transform of the product of the spatial transform of the derivative of the Green's function and the spatial transform of the input function will yield the solution to the optical field values at the observation plane. Equation 6 alone, or the combination of Equations 3 and 7, shows this notion in mathematical form.

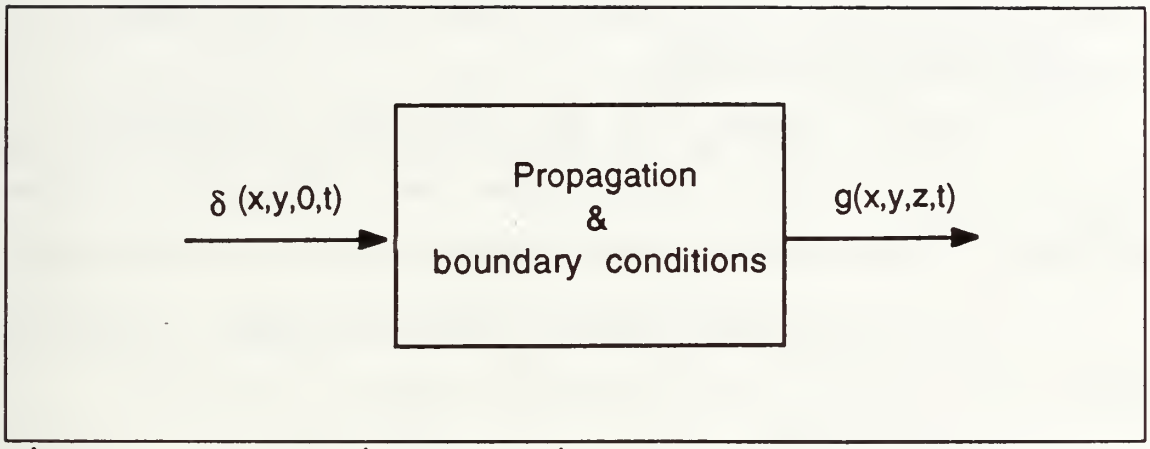

Figure 2. Block diagram of impulse response.

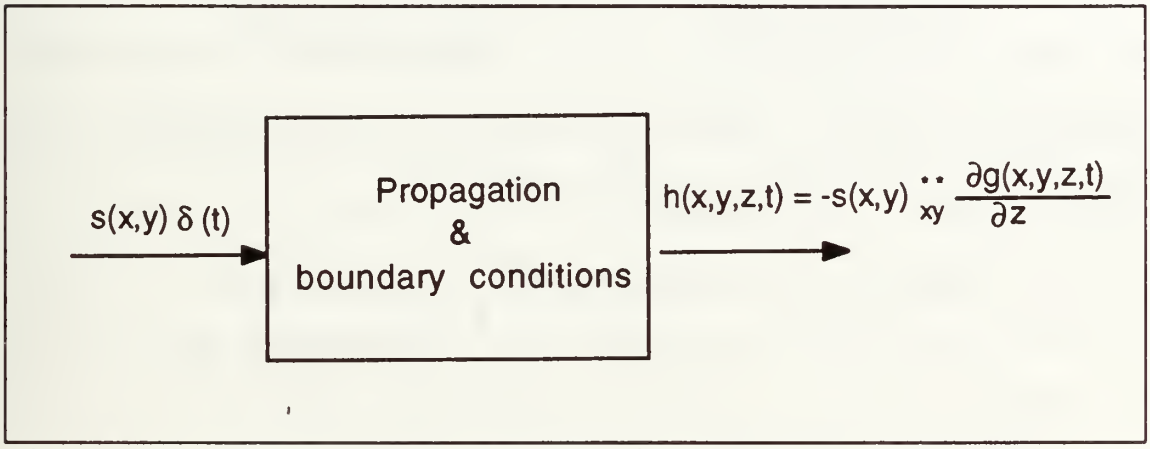

Figure 3. Block diagram of spatial impulse response.

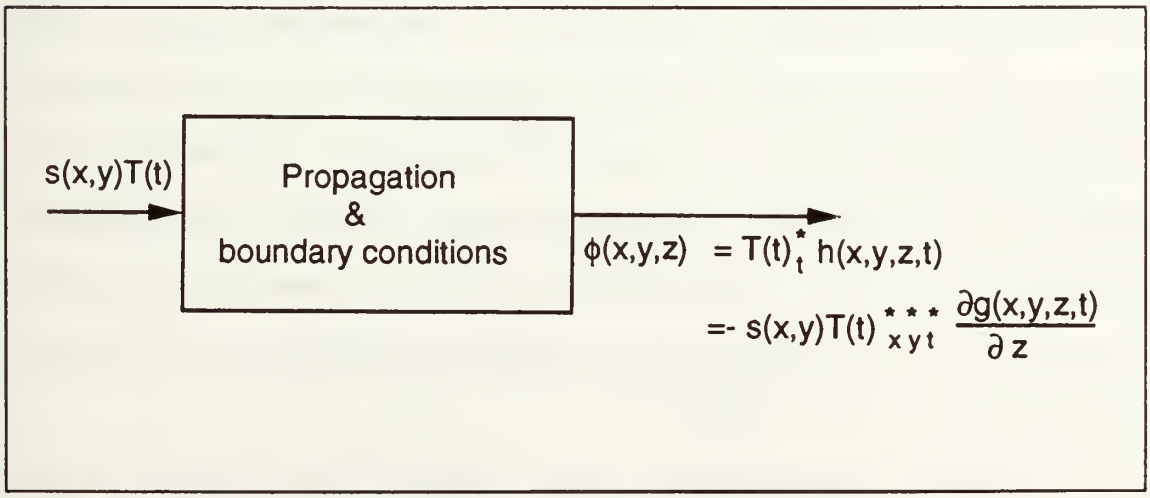

Figure 4. Block diagram of general solution.

C. MATHEMATICAL DERIVATION

The primary source for the following discussion is Guyomar's work in Ref. 4. Some few additions to the mathematics have been added and the explanatory text expanded.

The appropriate wave equation, again assuming a lossless propagation medium, is the Helmholtz equation given by

$$\nabla^2\phi - \frac{1}{c^2}\frac{\partial^2\phi}{\partial t^2} = 0. \tag{8}$$

The fields described by this wave equation are given by the radiation integral. However, assuming a planar aperture, the field $\phi(x,y,z,t)$ may be written [Ref. 4]

$$\phi(x,y,z,t) = \frac{\partial\phi_o(x,y,0,t)}{\partial n} \overset{*}{x}\overset{*}{y}\overset{*}{t} g(x,y,z,t) - \phi_o(x,y,0,t) \overset{*}{x}\overset{*}{y}\overset{*}{t} \frac{\partial g(x,y,z,t)}{\partial n} \tag{9}$$

where $\phi_o(x,y,0,t)$ is the excitation function. Its scalar amplitude distribution is known on the source plane. The Green's function satisfying the wave equation and the boundary conditions is $g(x,y,z,t)$ and the partial derivative with respect to n represents the normal derivative of the desired function for planar geometry. For an observation plane perpendicular to the z-axis, the normal derivative will become the derivative with respect to z.

In optical applications, the value of the field on the surface $\phi_o(x,y,0,t)$ is known. Therefore, it becomes desirable

to eliminate the first term of Equation 8 and to work solely with the second term. This can be done by using the Green's function which satisfies the condition that

$$g|_{z=0} = 0. \tag{10}$$

The Green's function that meets this condition is

$$g_- = \frac{\delta\left(t - \frac{\sqrt{r^2 + (z - z_0)^2}}{c}\right)}{\frac{\sqrt{r^2 + (z - z_0)^2}}{c}} - \frac{\delta\left(t - \frac{\sqrt{r^2 + (z + z_0)^2}}{c}\right)}{\frac{\sqrt{r^2 + (z + z_0)^2}}{c}}. \tag{11}$$

The "_" subscript associated with the g_- Green's function symbol simply indicates the sign between the two terms, where, if given different boundary conditions, a "+" would be more appropriate. Note, also, the conventional use of the letter "c" to indicate the speed of propagation. On the surface of the source plane at z=0, this Green's function has the following properties:

$$g_-|_{z=0} = 0 \tag{12}$$

$$\frac{\partial g_-}{\partial n}|_{z=0} = \frac{\partial g_-}{\partial z} \tag{13}$$

$$= -\frac{2z\,\delta(t - R/c)}{R^3} - \frac{2z\,\delta'(t - R/c)}{cR^2}, \tag{14}$$

where

Utilizing the Green's function g_-, the field can be written as

$$R = \sqrt{r^2 + z^2} \tag{15}$$

$$= \sqrt{x^2 + y^2 + z^2} \ . \tag{16}$$

$$\phi(x,y,z,t) = - \phi_o(x,y,z,t) \overset{*}{x} \overset{*}{y} \overset{*}{t} \frac{\partial g_-(x,y,z,t)}{\partial z} \ . \tag{17}$$

Substituting Equation 13 for $\partial g_-/\partial z$ results in

$$\begin{aligned} \phi(x,y,z,t) = {} & \phi_o(x,y,0,t) \overset{*}{x} \overset{*}{y} \overset{*}{t} \frac{2z\,\delta(t - R/c)}{R^3} \\ & + \phi_o(x,y,0,t) \overset{*}{x} \overset{*}{y} \overset{*}{t} \frac{2z\,\delta'(t - R/c)}{cR^2} \ . \end{aligned} \tag{18}$$

Since (f*g') = (f*g)' = (f'*g), the order of the derivative in the second term of Equation 18 may be interchanged, yielding

$$\begin{aligned} \phi(x,y,z,t) = {} & \phi_o(x,y,0,t) \overset{*}{x} \overset{*}{y} \overset{*}{t} \frac{2z\,\delta(t - R/c)}{R^3} \\ & + \phi_o'(x,y,0,t) \overset{*}{x} \overset{*}{y} \overset{*}{t} \frac{2z\,\delta(t - R/c)}{cR^2} \ . \end{aligned} \tag{19}$$

Previously, h(x,y,z,t) had been defined as the spatial impulse response (i.e., the linear response to an input which is multi-dimensional in space, yet an impulse in time). Recalling an excitation source separable in space and time has been assumed, this input is represented as

$$\phi_o(x,y,0,t) = s(x,y)\,\delta(t) \ . \tag{20}$$

Since the spatial impulse response is the convolution of the input function and the Green's function (refer to Fig. 3), the response may be expressed as

$$h(x,y,z,t) = s(x,y) \overset{*}{x} \overset{*}{y} \frac{2z\,\delta(t - R/c)}{R^3} + \delta'(t) \overset{*}{t} \left[s(x,y) \overset{*}{x} \overset{*}{y} \frac{2z\,\delta(t - R/c)}{cR^2} \right] . \tag{21}$$

Rather than attempting to perform the spatial convolutions over x and y, the product shall be taken following the two-dimensional spatial Fourier transform of Equation 21. This transform results in

$$\tilde{h}(f_x, f_y, z, t) = \frac{\tilde{s}(f_x, f_y)\ 2zJ_0(\rho\sqrt{c^2t^2 - z^2})}{c^2t^2} + \delta'(t) \overset{*}{t} \left[\frac{\tilde{s}(f_x, f_y)\ 2zJ_0(\rho\sqrt{c^2t^2 - z^2})}{c^2t} \right] \tag{22}$$

where $\tilde{h}$ is the two-dimensional spatial transform of h, f_x and f_y are the spatial frequencies, ρ is the radial spatial frequency ($=\sqrt{f_x^2 + f_y^2}$), and the transform pair [Ref. 4]

$$F_{xy}\left\{ \frac{\delta(t - R/c)}{R^n} \right\} = \frac{J_0(\rho\sqrt{c^2t^2 - z^2})}{(ct)^{n-1}} \tag{23}$$

has been utilized.

Recognizing that time convolution with the time derivative of the delta impulse is identical with taking the derivative of the function in the time domain, i.e.,

$$f(t) \overset{*}{t} \delta'(t) = f'(t) , \tag{24}$$

the spatial impulse response can now be written as

$$h(x,y,z,t) = \frac{2z}{c^2t^2}F^{-1}\left\{ \tilde{s}(f_x,f_y)\,J_0(\rho\sqrt{c^2t^2 - z^2}\,)H(t - z/c)\right\} + \frac{\partial}{\partial t}\left[\frac{2z}{c^2}\,F^{-1}\left\{\frac{\tilde{s}(f_x,f_y)\,J_0(\rho\sqrt{c^2t^2 - z^2}\,)H(t - z/c)}{t}\right\}\right] \quad (25)$$

The expression H(t-z/c) is the Heaviside step function which indicates that no output is possible until the propagation wave has travelled the distance from the source plane to the observation plane. This function assures causality at the observation plane. Focusing on the second term of Equation 25, the partial derivative with respect to time, being independent of the transform variable, is moved through the transform so that the spatial impulse response, in its final form, is expressed as

$$h(x,y,z,t) = \frac{2z}{c^2t^2}F^{-1}\left\{ \tilde{s}(f_x,f_y)\,J_0(\rho\sqrt{c^2t^2 - z^2}\,)H(t - z/c)\right\} + \frac{2z}{c^2}F^{-1}\left\{ \tilde{s}(f_x,f_y)\,\frac{\partial}{\partial t}\left[\frac{J_0(\rho\sqrt{c^2t^2 - z^2}\,)H(t - z/c)}{t}\right]\right\} \quad (26)$$

Viewing Equation 26, the calculation of the temporal impulse response proceeds as follows:

1. The excitation function $\phi_0(x,y,0,t)$ is separated into its components s(x,y) and T(t), i.e., $\phi_0(x,y,0,t) = s(x,y)T(t)$.

2. The distance z to the observation plane is chosen and a value of time t is also chosen, where t is greater than or equal to z/c.

3. Calculate the two-dimensional spatial transform of s(x,y).

4. To calculate the first term on the right side, multiply the transform by $J_0(\rho\sqrt{c^2t^2 - z^2})$.

5. Perform a two-dimensional inverse-transform on the product and multiply the result by the constant $2z/(c^2t^2)$.

6. The second term requires several operations prior to the inverse transform being performed. First, divide the Bessel function by the time t. Next, the time derivative is taken and the result is multiplied by the two-dimensional spatial transform of s(x,y). At this point the inverse transform is performed and its result is multiplied by the constant $2z/c^2$.

7. Find the sum of the two terms, and thus the process is complete for a single moment in time. Time may be incremented as desired, and the process repeated from step 3.

Within the transform domain, the propagation equation may be viewed as being composed of two time-varying spatial "filters". The first term of Equation 26 shows the transformed excitation function being multiplied by a Bessel function whose argument varies with time. Figure 5 illustrates two examples of the filter at different times. The filter is, in effect, collapsing in on itself as the

argument grows larger. Additional examples of this filter are shown in Appendix A.

The partial derivative of the second term acts in a similar vein. Pictorially, as seen in Fig. 6, it begins to appear somewhat similar to that of the Bessel function with the obvious difference that the center of the function at $\rho=0$ is no longer the peak value. Additional examples of this are shown in Appendix B.

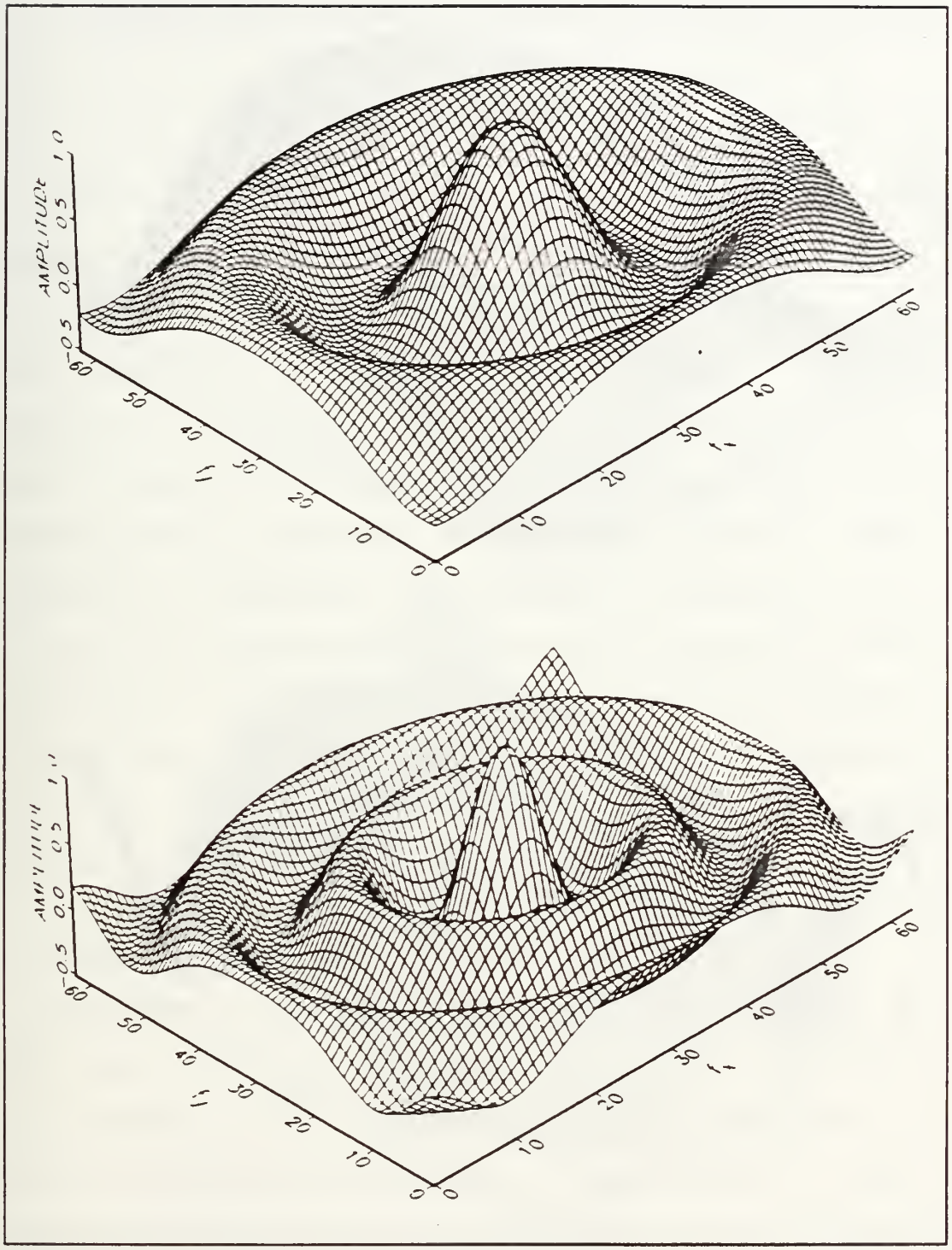

Figure 5. Time-varying Bessel filters at two values of time.

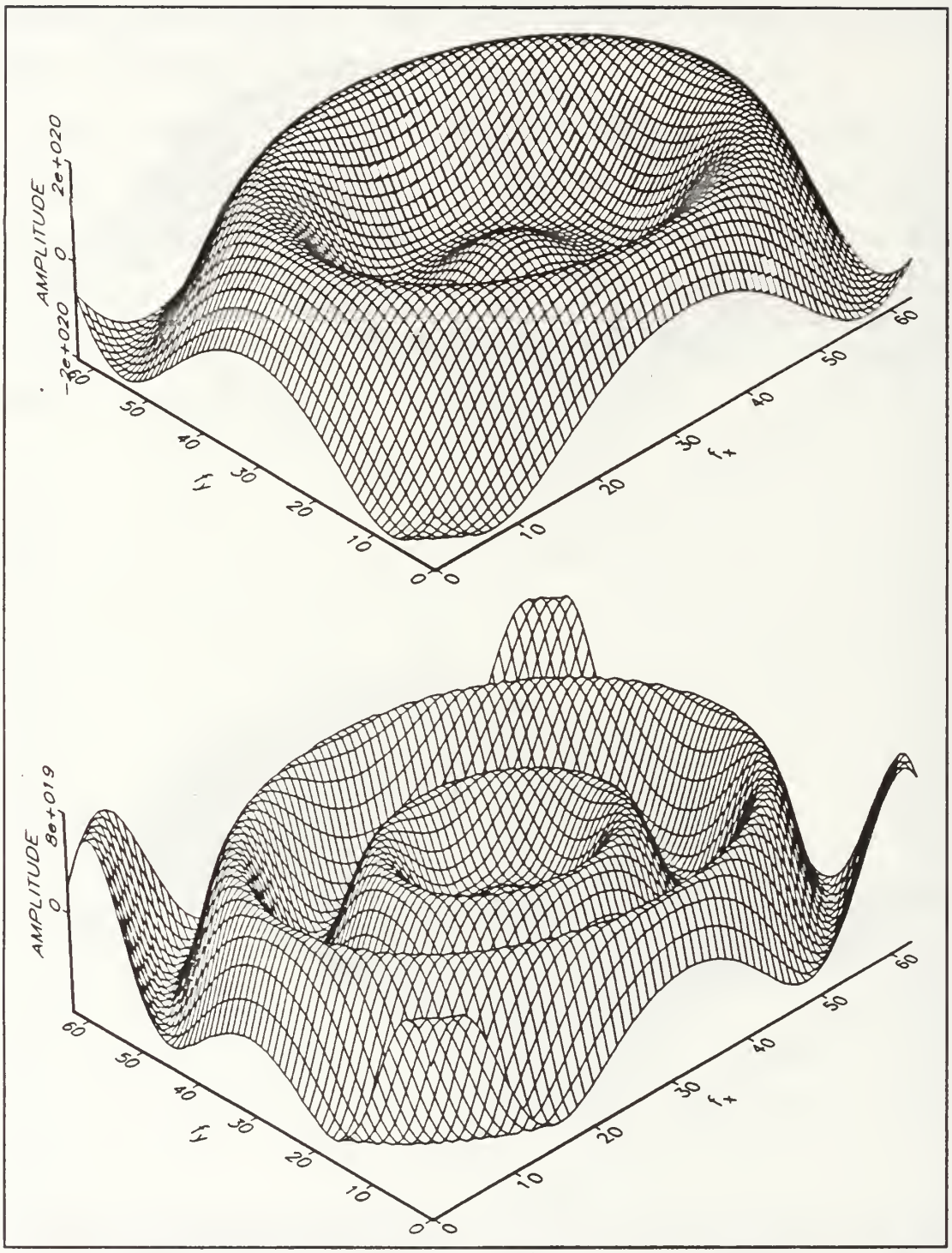

Figure 6. Time-varying derivative filters at two values of time.

III. PROGRAM DESCRIPTION

This chapter addresses the formulation of the MATLAB program written to model scalar optical wave propagation. No in-depth knowledge of MATLAB is assumed and the discussion of the program is as "functional" as possible. A detailed explanation of the code is found is Appendices C and D. Specific aspects of MATLAB which drove decisions concerning the program is presented in the first section. This is followed by a discussion of the program's modularity, which is of significant importance. A functional discussion of each of the program's two modules in separate sections completes the presentation. Many figures are incorporated throughout the chapter and they are found at the end of each applicable section.

A. OVERVIEW OF MATLAB

MATLAB is an interactive software package developed by The Mathworks, Inc. of Natick, MA. Its specific orientation is for scientific and engineering numeric computations, and problem solutions are expressed almost exactly as they are written mathematically. As a result, many of the frustrations and time-consuming development processes of conventional programming are avoided. The name MATLAB stands for matrix

laboratory, and its basic data element is a matrix which does not require dimensioning. Numerical analysis, matrix computation, signal processing, and a graphics capability have all been integrated. The user is also able to incorporate self-developed functions, as has been done for the various excitation functions. [Ref. 11]

Of critical importance to this project were the intrinsic functions of two-dimensional FFT's (FFT2), Bessel function of the first kind of order zero calculations, and 3-D graphics. The Matlab User's Guide [Ref. 10] goes into some detail in explaining the specific algorithm called by MATLAB's different functions. It points out that when the row/column dimensions of the matrix are a power of two, a high speed radix-two FFT algorithm is used. When the dimensions are not even, a non-power-of-two algorithm finds the prime factors of the dimensions and computes the mixed-radix discrete Fourier transform. It cautions that this latter process can become quite time consuming, particularly as the size of the matrices become larger. For this reason, the decision was made to work with NxN matrices, where N is an even number.

MATLAB employs a backwards three-term recurrence equation to calculate each value of the Bessel function when the order is an integer. This routine is time consuming, given the size of the matrices required and the number of values to be found. Therefore, the program execution has been "modularized" in the sense that these Bessel calculations are accomplished

separately, and the results, which are a function of radial distance and time (see Equation 26), are stored to disk to be called when the second module is executed. This has the advantage of allowing a different excitation function to be analyzed without recalculating the Bessel values. This will be also be discussed futher in an upcoming section.

Additionally, MATLAB has no built-in function to calculate the derivative of a multi-dimensional function. To complete such an operation requires approximating the derivative through a difference operation. The approach taken will be explained in detail in a following section.

MATLAB's on-line graphics capability allows viewing outputs immediately, rather than having to transport the data into a separate graphics program. Unfortunately, MATLAB does not numerically scale the axes when plotting the data in 3-D. Hence, it is useful for comparing relative shapes, but an alternative graphics program must be found if absolute amplitudes are to be displayed. All subsequent figures have been generated on a program called AXUM, produced by TriMetrix, Inc. of Seattle, Wa. AXUM does have scaled axes, along with many other features, some of which will be briefly described in Chapter IV.

B. DISCUSSION OF PROGRAM MODULARITY

As previously pointed out, the program has been "modularized" in an effort to separate the time-consuming

calculation of the Bessel filter from the remainder of the operations.

The first module has been named OPTFIL for OPTical FILter. It could just as easily have been designated BESSFIL for the time-varying Bessel filter which it generates, but work is also being done for an acoustic case, and the name provides a simple means of differentiating between the two.

The second module completes the solution of the optical field at the observation plane as it is defined by Equation 26. This code, named OPTPROP for OPTical PROPagation, provides for a selection of a specific spatially-distributed excitation function, transforms it via the FFT2 procedure, generates the derivative type filter, performs the multiplications with each of the filters, inverse transforms the results, multiplies each term by its respective constants, and, lastly, adds the two terms together for the output. Thus, the output over the entire observation space is computed. However, only data from the center row of the resultant matrix (i.e., h(0,y,z,t)) is used to construct the final output presentation. Each time increment builds this center row information sequentially as columns of the output data matrix. Such a display conforms with the output plots most commonly seen in the acoustics literature.

C. FUNCTIONAL EXPLANATION OF OPTFIL

OPTFIL, in order to generate the time-varying Bessel filter, must initially define the elements of the argument of the Bessel function. These elements are "ρ", the spatial radius of the filter; "c", the velocity of propagation (3E8 meters/second); "t", the incremental value of time; and "z", the distance between the source plane and the observation plane.

Of these, t requires additional stipulations since it is incrementing. The Heaviside step function, H(t-(z/c)), is simulated by commencing time at (z/c). The output plot, it was decided, would illustrate the causality at the observation plane by displaying a small number of initial time increments with zero output. In the program this arbitrary number of increments is defined as the variable "Step". Some maximum time must be specified as well, and it is called "time_max". The total number of time increments between the values of "time zero" and time_max is expressed as the variable "M". The value of M includes the number of increments specified in Step, and so (M-Step) time "slices" are calculated between z/c and time_max. An additional time-related variable is "t_eps". This was named for a time_epsilon added to each incremental time slice for the purpose of calculating the derivative spatial filter.

Some variables are utilized solely within OPTFIL; others are needed by both OPTFIL and OPTPROP. Those in the latter

category are stored in a separate file called OPTBES which is saved to the hard drive to be reloaded when OPTPROP begins executing.

The dimension of the transform space and the calculation of the variable "rho" are missing from the explanation thus far. The transform space is expressed in terms of an "NxN" square matrix. Recall that to facilitate the FFT2 calculation, the matrix was to be of even dimensions. Because MATLAB matrix indices begin with 1 rather than 0 (the upper left entry being row 1, column 1 not row 0, column 0 as an origin would require), a matrix of dimension NxN will have N points and N-1 segments in each row and column. At first glance, the center of symmetry of the array would be at the (N+1)/2 row and the (N+1)/2 column. But the use of an even number of points means that this center of symmetry does not have an array point associated with it. Therefore, the center of the transform space, the variable "N0", was taken to be at the position ((N/2)+1, (N/2)+1). For a 64x64 matrix, then, the center is located at position (33,33) and spatial frequency radius rho is measured radially outward from that point. This forced arrangement where the defined center is not the geometric center is illustrated with a circular function in Fig. 7.

A Fourier Transform (or the inverse transform) performed on such a function with a displaced center of symmetry would calculate the amplitude correctly but the phase would be

incorrect. This problem with the phase calculation can be overcome by reorienting the function into a corner geometry, as shown in Fig. 8. Here one can see the function centered at what would be the (0,0) position, given a quadrant I perspective. The FFT2 and IFFT2 algorithms within MATLAB assume a replicated image beyond the matrix boundaries so that mathematically the symmetry of the function is retained. The transform of the re-oriented function has the correct amplitude and phase.

Figure 9 depicts a 64x64 array base situated on the x,y plane divided into four quadrants. The notion of the quadrants is important because the matrix generated by OPTFIL is symmetric in rho with respect to the center of symmetry. As a result, the calculation for only one quadrant need be done and flipping the data appropriately fills the remaining quadrants. This actually becomes somewhat tricky because of the fact that the quadrants are unequal in size. Quadrant II consists of rows 1 through 33, and columns 1 through 33 for a 33x33 "subarray", while quadrant I includes rows 1 through 33, and columns 33 through 64 for a 33x32 "subarray". Quadrant III is different still with dimensions 32x33, and quadrant IV is the smallest of the four as a 32x32. Again, the different sizes result from the designation of element (33,33) as the center of the transform space.

Calculation of rho is done in a straight-forward manner due to a MATLAB function called "meshdom". This constructs a

matrix of an x,y grid incremented as specified out to a specified distance. The grid elements in the program are named "rhox" and "rhoy". Rhox and rhoy are the cartesian sides of the radial length rho. Rho was arbitrarily defined to be 200 frequency units in length for N0-1 points, and a vector called "rho_m" was constructed with N0 points so that the largest quadrant could be properly sized. Rho_m is the input value for the meshdom function and, thereby, an N0xN0 sized matrix is created. Applying the Pythagorean theorem to rhox and rhoy, rho at element (33,33) is zero; at element (33,64), it is 200; and, at elements (33,1) and (1,33), it is approximately 206.45. The maximum value of approximately 291.96 is found at (1,1), located in the largest of the quadrants, as this location is the farthest point from the center at (33,33).

To continue with the calculation of the Bessel filter, the rho value of each element in the x,y grid whose points are defined by (rhox, rhoy) is multiplied by a constant $\sqrt{c^2t^2-z^2}$, to form the argument for the Bessel function. After the Bessel operator has been invoked for each element, the results are contained within a temporary matrix called "TEMP". TEMP is then flipped to fill in the quadrants of the NxN sized transform space. The calculated NxN matrix is called PROP. A numerical suffix is added to PROP to correlate the data with the time index, e.g., PROP1A, PROP5B. The additions of "A" and "B" serve to differentiate between those matrices computed

using the values of time contained within the time vector and those to which t_eps has been added to the time values. For example, PROP22A is computed using the twenty-second value of time in the time vector. PROP22B is computed using that same time value plus t_eps.

The variable which changes from iteration to iteration is time. Recall that the data commences with time z/c and continues to time_max. The output PROP matrix for each value of time is saved to disk contained within a file named "p(N)x(time index)(A/B)". A numerical suffix is added to "p" to delineate the size of the matrix (N) and to identify the time slice from which it was formed. For example, file names are of the form p64x10A or p128x45B. In these two examples, the matrices are of dimension 64x64 and 128x128, respectively. The former contains the output matrix PROP10A and the latter contains the output matrix PROP45B.

Examples of the Bessel filters, taken at time slices (1), (3), (8), and (23) are illustrated in Figs. 10-13. The planar appearance of the filter at time (1) is due to the fact that the value of time is equal to z/c and the argument of the Bessel function is zero. In each of the remaining illustrations, the maximum value of 1 is found at the center point of the transform space where the radial distance rho is equal to zero. The symmetry property is easily seen, as is the progression of the filter "collapsing" in on itself as

more and more peaks continue to be formed. Additional examples of the filter are contained in Appendix A.

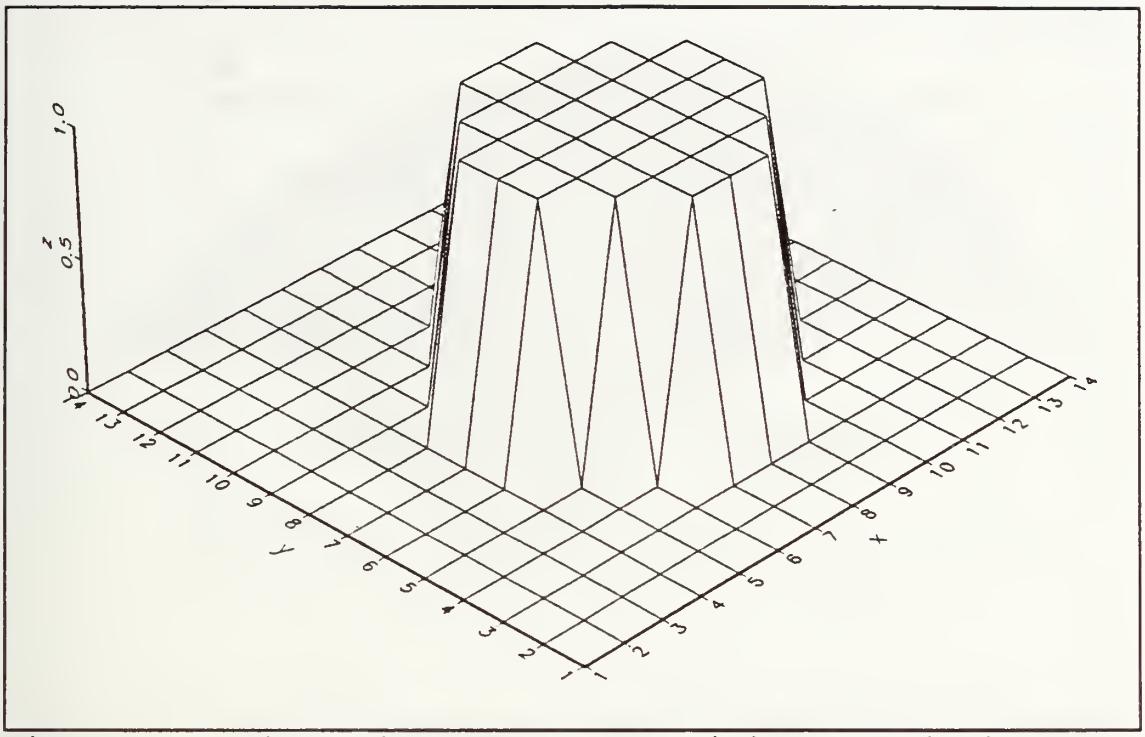

Figure 7. Object with off-center origin (for viewing).

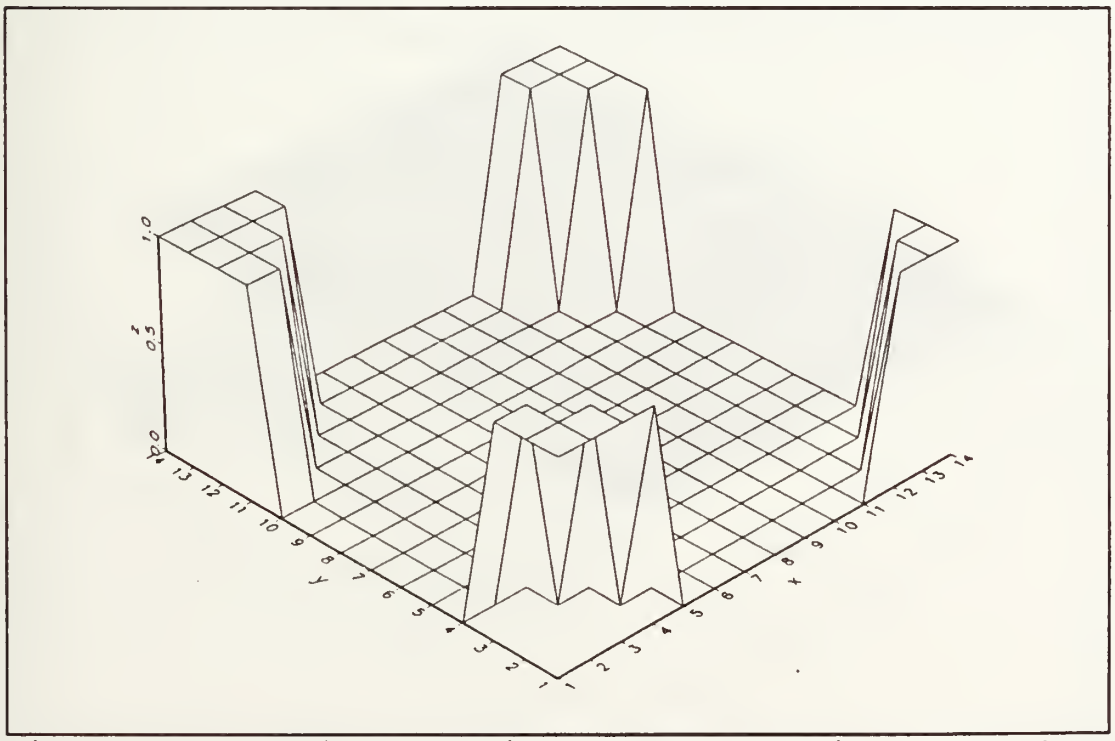

Figure 8. Shifted object (for Fourier transform calculation).

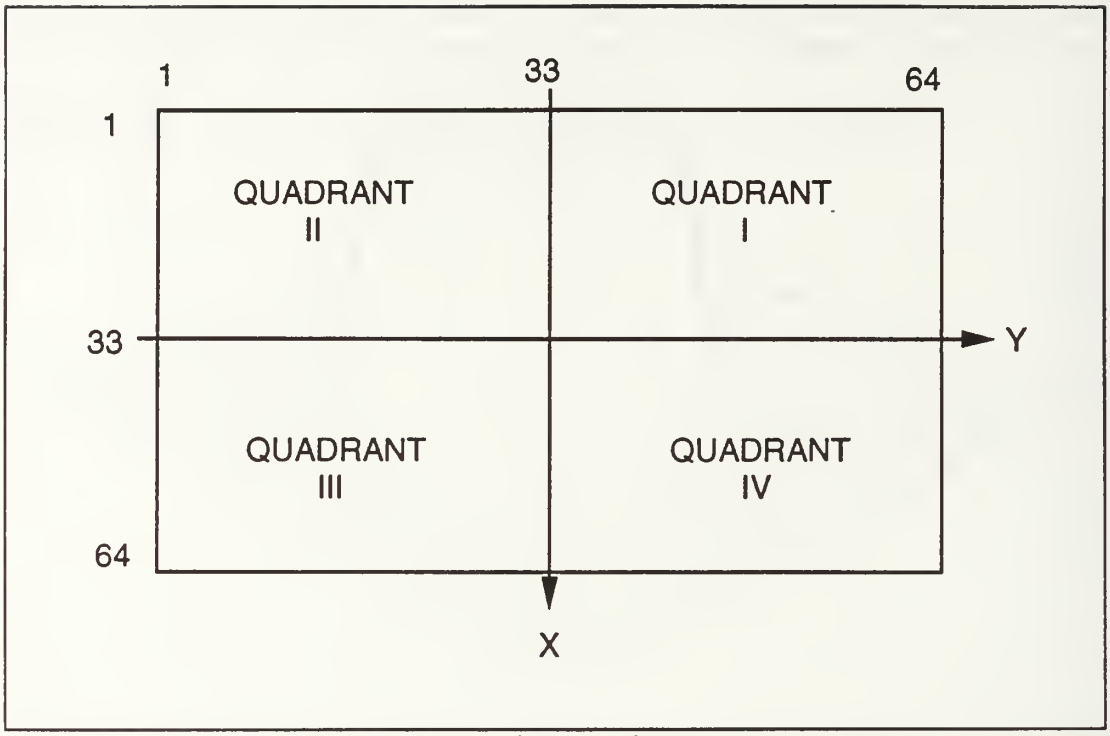

Figure 9. Base array configuration.

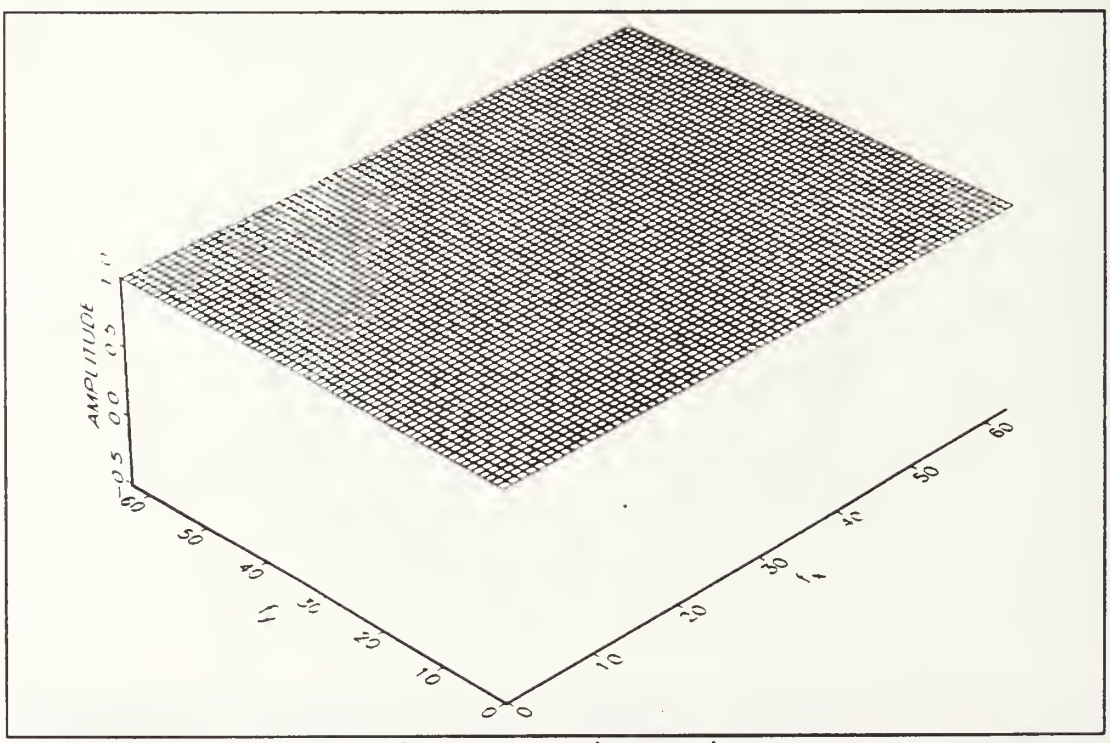

Figure 10. Bessel filter at time slice 1.

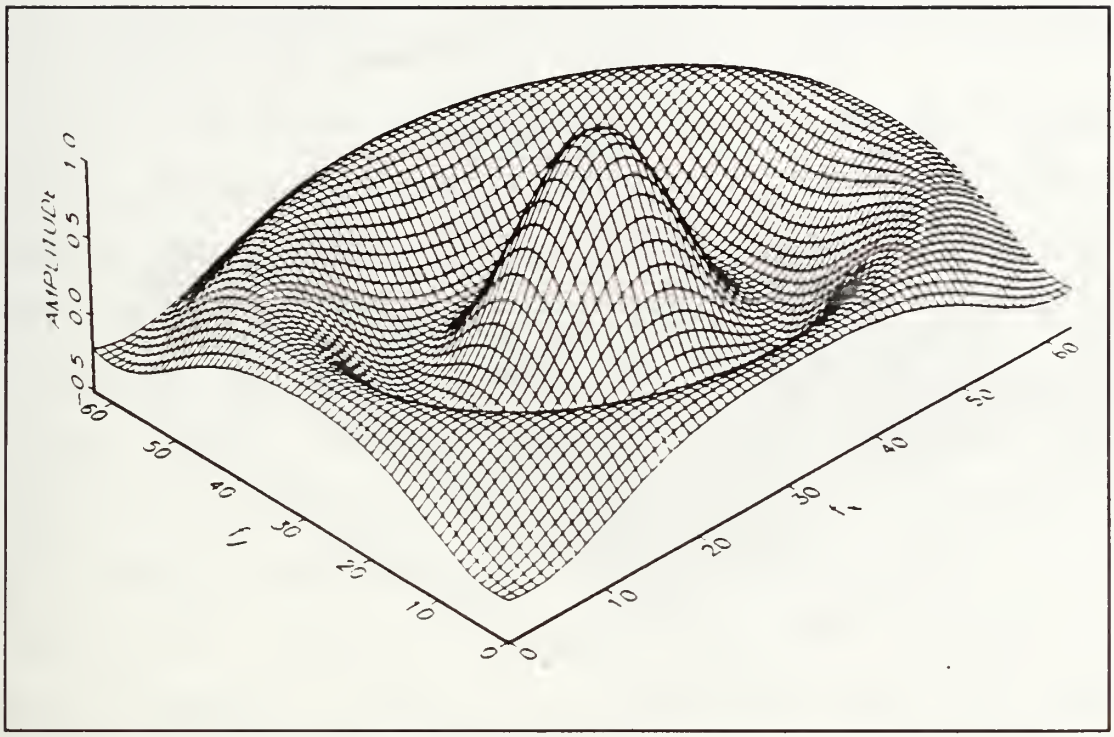

Figure 11. Bessel filter at time slice 3.

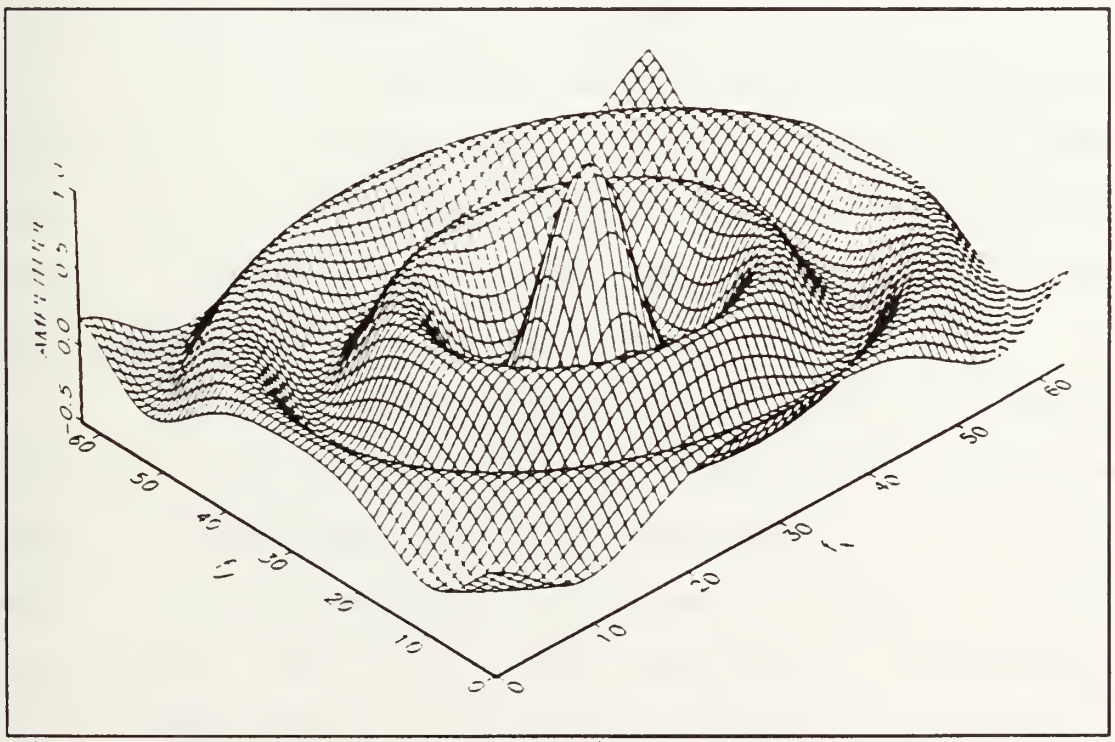

Figure 12. Bessel filter at time slice 8.

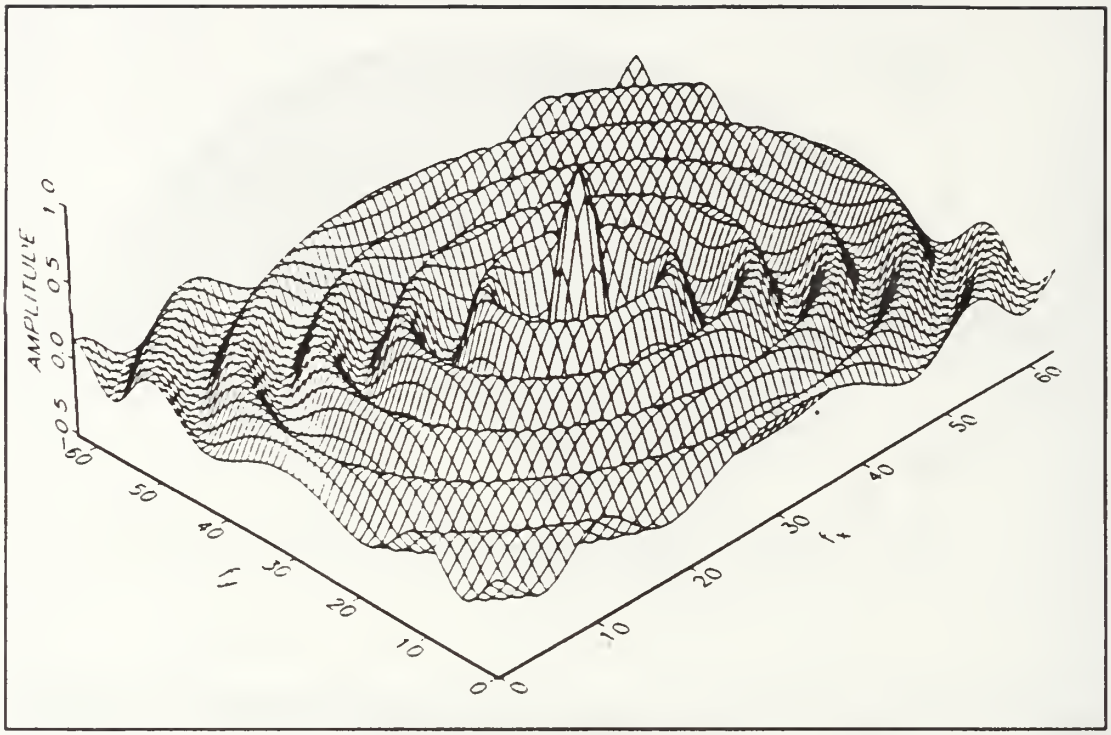

Figure 13. Bessel filter at time slice 23.

D. FUNCTIONAL EXPLANATION OF OPTPROP

This discussion of the second program module, OPTPROP, will be divided into three sections. The first addresses the input variables and functions. The second section will be a functional walk-through of the code, with many illustrations to amplify the text. These figures reflect the calculations at time (1), i.e., only at time z/c. With an understanding of the variable names and the process, the third section will then provide discussion and illustrations of the various computational steps as time progesses. The specific time slices selected for analysis are those for which the Bessel filter results were portrayed-- time (1), (3), (8), and (23). All figures are incorporated at the end of each sub-section.

1. Input variables and functions

The initial operation performed by OPTPROP is to load from disk the variables utilized by both OPTPROP and OPTFIL. Among other things, these variables establish the size of the matrix, define the center point, and set the value of Step. The entire time vector is also passed.

OPTPROP provides screen text to allow a keyboard selection of any of the four possible excitation functions. These are named Circle, Table, Circular Gaussian, and Circular Bessel in accordance with their respective spatial amplitude distribution. The final two were developed for future efforts in this study, and no illustrations nor final outputs will be

shown for them. Once the input function has been chosen, the program asks for either the diameter or the width, as appropriate. Default values are shown in brackets. These width values must be odd numbers for the input algorithms to operate properly, and the screen text includes a reminder of that.

The input generated was viewed via the built-in MATLAB graphics. An example of a Circle with radius 17 in a 64x64 matrix is shown in Fig. 14. This is the input used in all the following illustrations in this chapter, Appendix A and Appendix B.

On the MATLAB display, the title "SHFT_INPUT" will also be included. Any illustration, whether in the time domain or the transform domain which has its peak in the center of the viewing quadrant is preceded by a SHFT modifier. This is to differentiate that configuration from the one in which preparation for the FFT2 operation has been done. The corner geometry previously mentioned is shown for this Circle input in Fig. 15. It, like every output which follows in the program, can also be viewed; its title, due to the corner geometry, is simply "INPUT". Once it has been transformed by FFT2, it is then called "F_INPUT" for the frequency domain. This is shown in Fig. 16. If shifted back to the center configuration still within the frequency domain, the title is "FSHFT_INPUT". See Fig. 17. Only the shifted results will be shown from this point on.

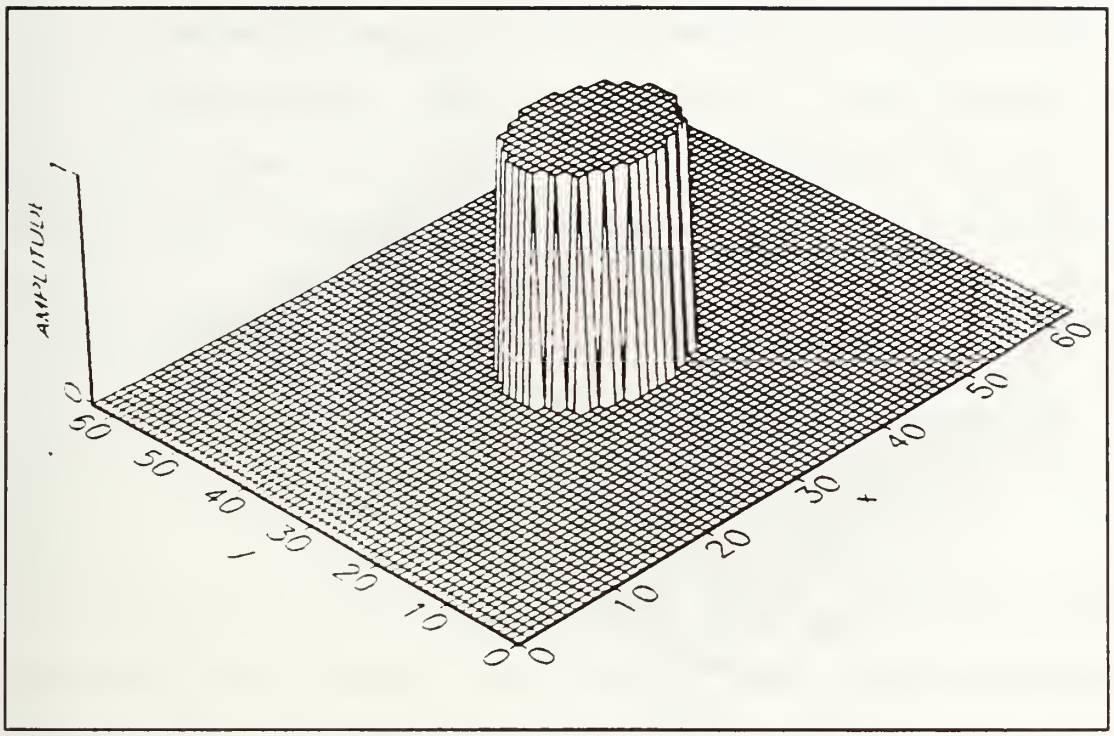

Figure 14. SHFT_INPUT (input in center geometry).

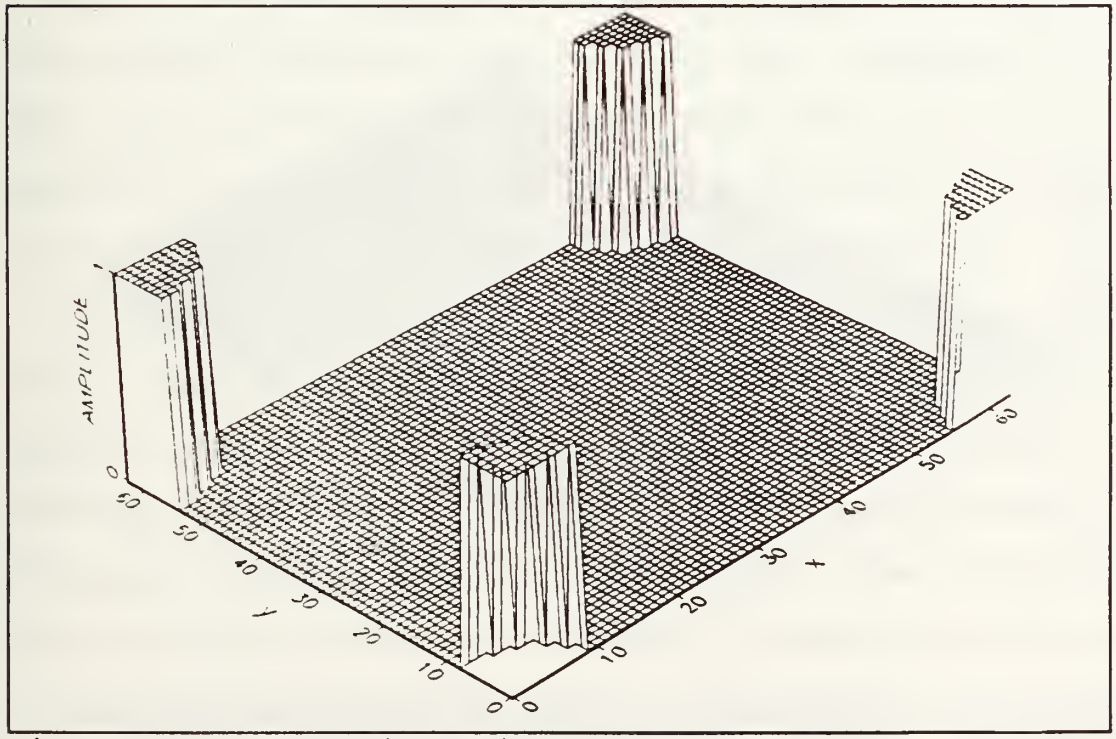

Figure 15. INPUT (input in corner geometry).

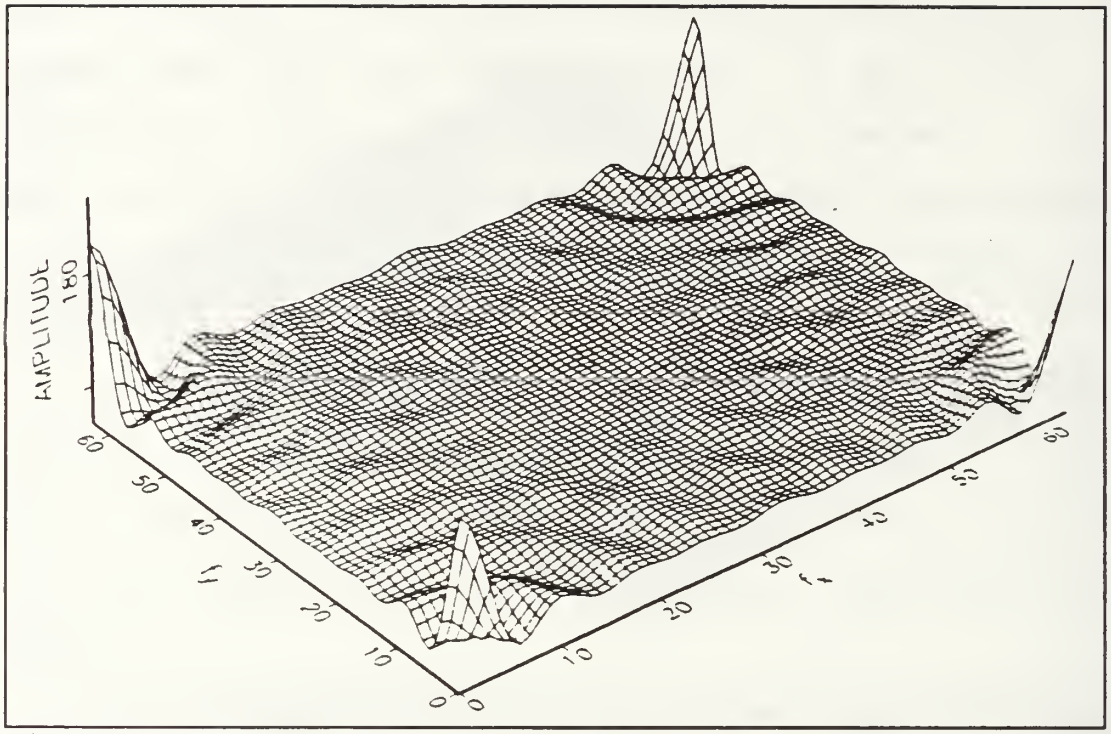

Figure 16. F_INPUT (transformed input in corner geometry).

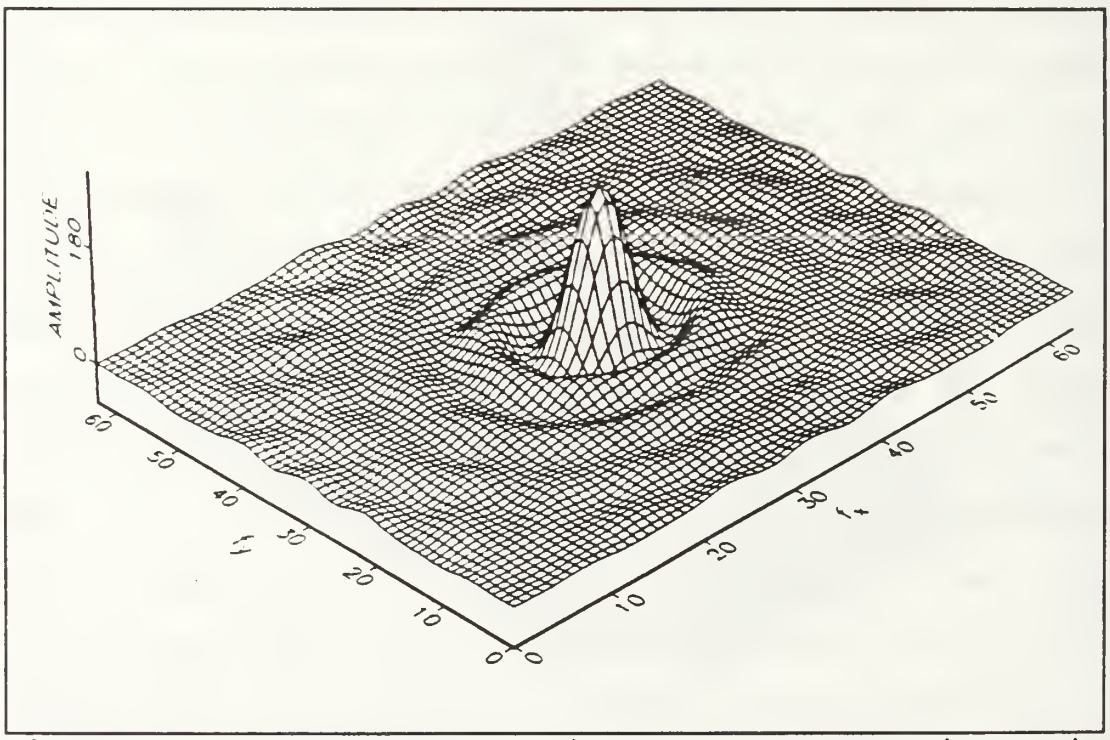

Figure 17. FSHFT_INPUT (shifted transformed input in center geometry).

2. View at a single point in time

At this point, the program begins the loop through the values of time to perform the calculations required by Equation 26. The equation is repeated below for reference.

$$h(x,y,z,t) = \frac{2z}{c^2t^2}F^{-1}\left\{ \tilde{s}(f_x,f_y)\,J_0(\rho\sqrt{c^2t^2 - z^2}\,)\,H(t - z/c)\right\} + \frac{2z}{c^2}F^{-1}\left\{ \tilde{s}(f_x,f_y)\,\frac{\partial}{\partial t}\left[\frac{J_0(\rho\sqrt{c^2t^2 - z^2})\,H(t - z/c)}{t}\right] \right\} \quad (27)$$

The A and B pairs of PROP matrices, containing the Bessel filter data, are loaded from the disk into the RAM workspace by calling the appropriate files, and are renamed "vname1" or "vname2", respectively. Recall that the file names with an A addendum to p(N)x(time index), such as p64x1A, contain the PROP matrices formed by the value of time contained in the time vector at that index. Those PROP matrices from files appended with a B utilize the time value resulting from adding t_eps to the value contained within the time vector.

The first term of the equation given above, called "Fshft_output1" in the program, is formed by the element-by-element product of vname1 and Fshft_input. Fshft_input is necessary rather than F_input since the PROP matrix peak is in the center oriented geometry. (One or the other of PROP or F_input required adjustment so that the geometry was alike and F_input was arbitrarily selected.) Fshft_output1 for time (1)

is shown in Fig. 18 (all figures shown in this sub-section reflect time slice (1)). In this instance, Fshft_output1 is identical to Fshft_input because the Bessel filter is planar with a value of 1. Figure 10 shows that Bessel filter.

Calculation of the second term was complicated by the fact that MATLAB does not contain a function for the derivative of a multi-dimensional object such as the Bessel filter. For that reason, a difference approximation was adopted. The equation given below illustrates the method. The index of the value of time is denoted in the equation by "m".

$$\frac{\frac{PROP(m)B}{t(m)+t_{eps}} - \frac{PROP(m)A}{t(m)}}{t_{eps}} \qquad (28)$$

In words, each PROP matrix of the PROP A/B pair is divided by its respective time. PROPA is subtracted from PROPB and the result is divided by the time difference between the two, which is t_eps. Admittedly, this is a difference approximation to the derivative, and further work could be done in arranging a "two-sided" derivative, where plus and minus t_eps are applied to a given reference PROP matrix. This portion of the second term which contains the derivative operation is called within the program Der_fil for DERivative FILter. An example of this filter is shown in Fig. 19. Additional examples are contained in Appendix B. The product

of Der_fil and Fshft_input forms Fshft_output2, which is shown in Fig. 20.

In preparation for the IFFT2 operation, both Fshft_output1 and Fshft_output2 are converted to the corner geometry and the inverse transform is then performed. Each is multiplied in the same coding step by its respective constant to form "output1" and "output2". "Shft_output1" and "shft_output2" are formed prior to the summation step which forms "shft_output". Each of these shifted versions is shown in Figs. 21-23. Shft_output is not what will be actually observed since optical detectors are square law detectors. To account for this, the program takes the magnitude of shft_output to form "shft_outabs". This final output for time (1) is shown in Fig. 24.

Of this output matrix, only the N0 row, h(0,y,z,t), is saved for inclusion in the plot which will depict the field over time. The information is saved to one line of a matrix called "output_plot". The final form of the matrix has, following "Step" columns of zeros, the results from the center of each solution matrix for the successive values of time in successive columns. Thus, the final output plot shows the field distribution of only the center line of the observation plane's aperture over time, although the time-changing field over the entire aperture has been calculated. Depiction of the final output plot is reserved until Chapter IV. This concludes the second section of the explanation of OPTPROP,

and what follows is a look at the solution steps previously discussed as they appear over time.

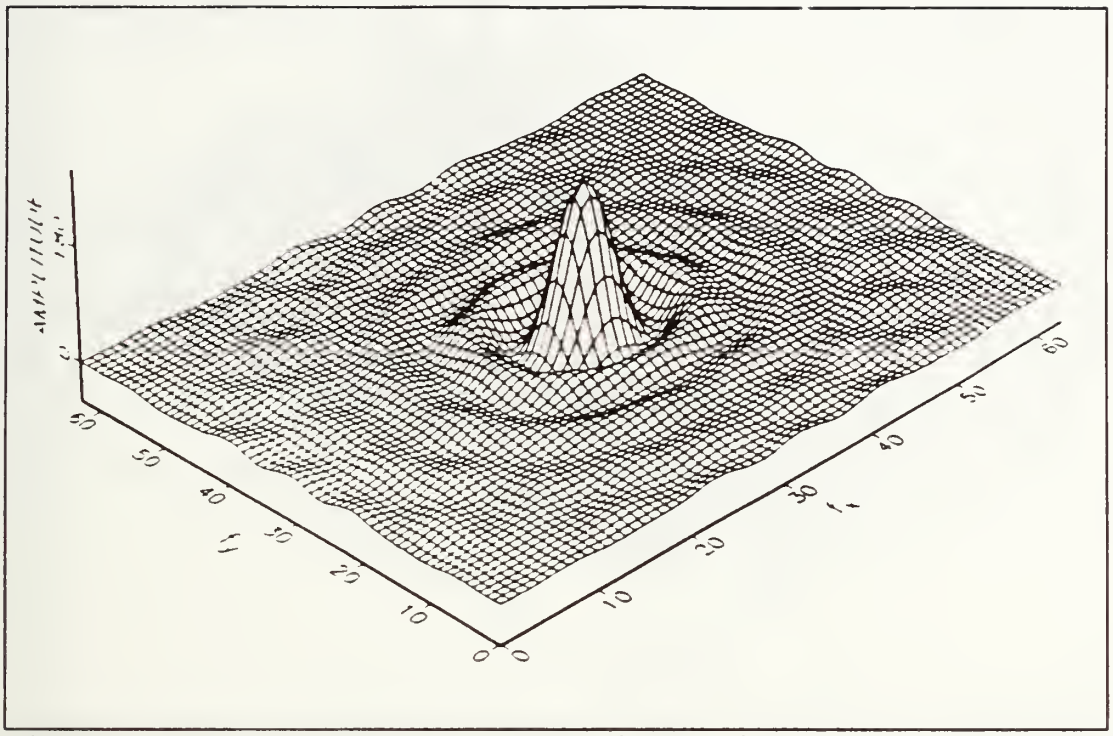

Figure 18. FSHFT_OUTPUT1 at time slice 1 (product of Bessel filter and transformed input in transform domain).

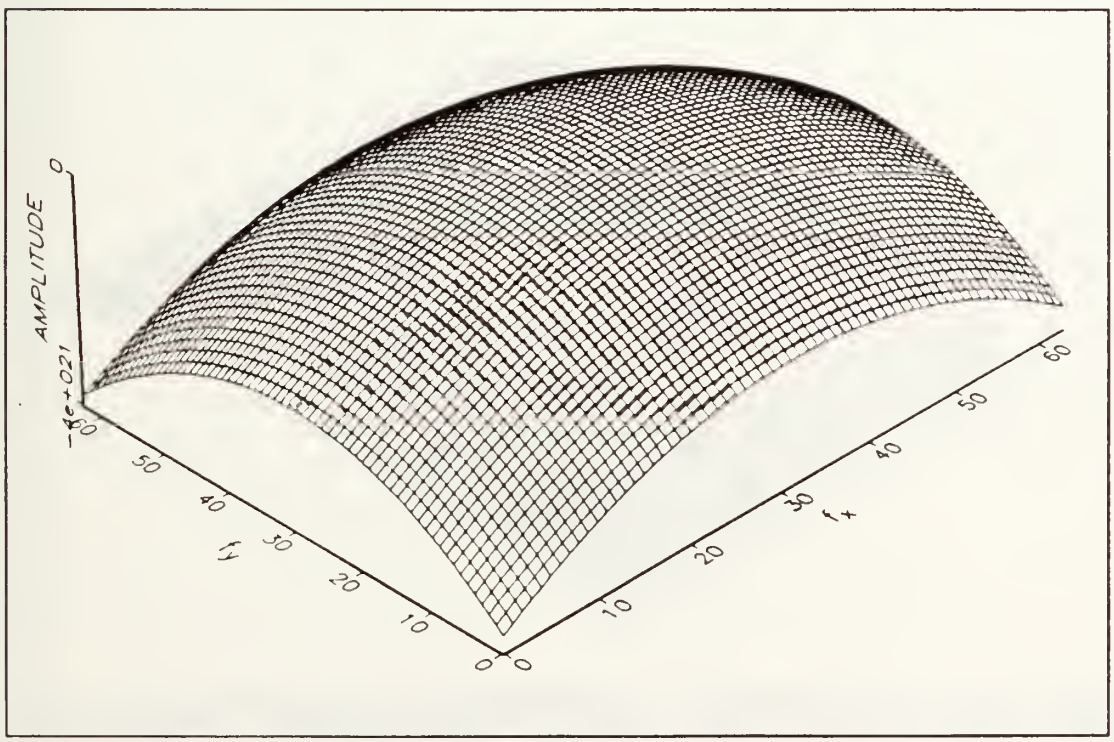

Figure 19. DER_FIL (derivative filter).

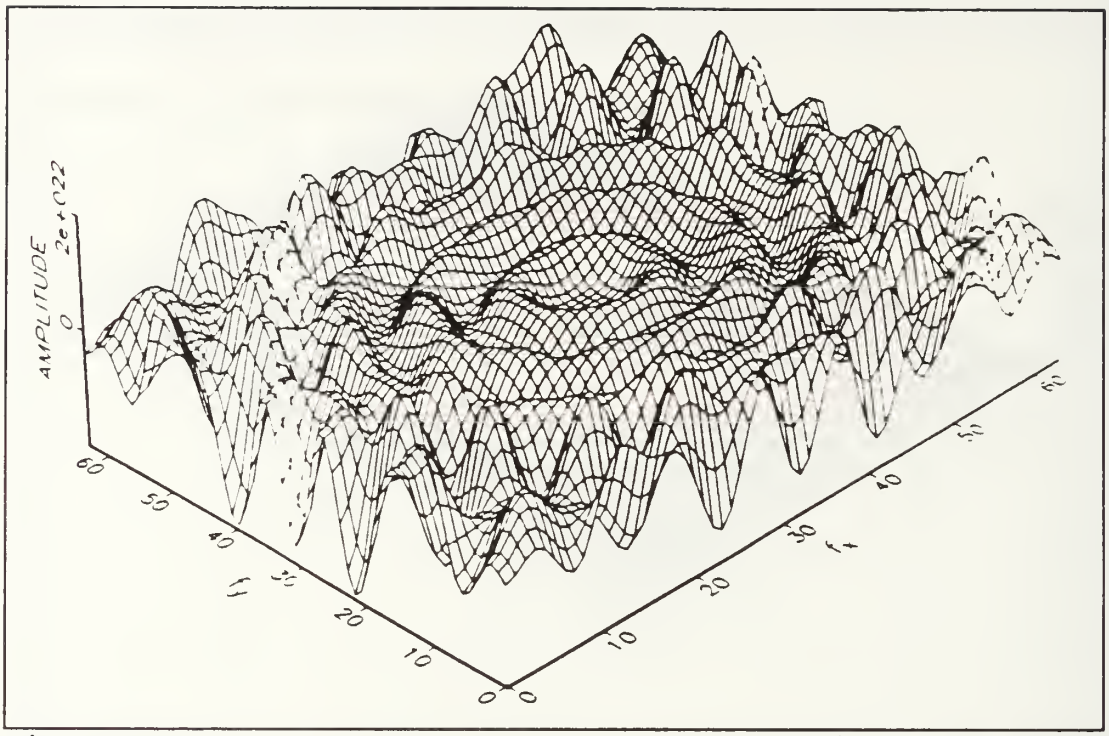

Figure 20. FSHFT_OUTPUT2 (product of derivative filter and transformed input in transform domain).

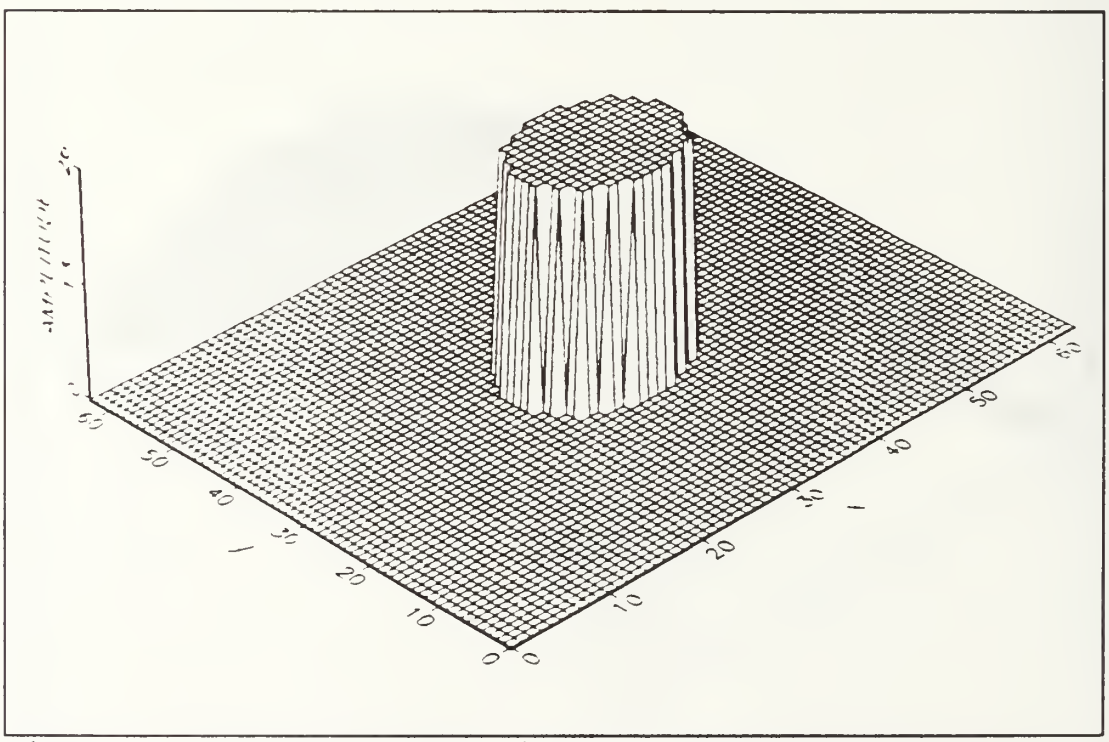

Figure 21. SHFT_OUTPUT1 (inverse transform of product of Bessel filter and transformed input).

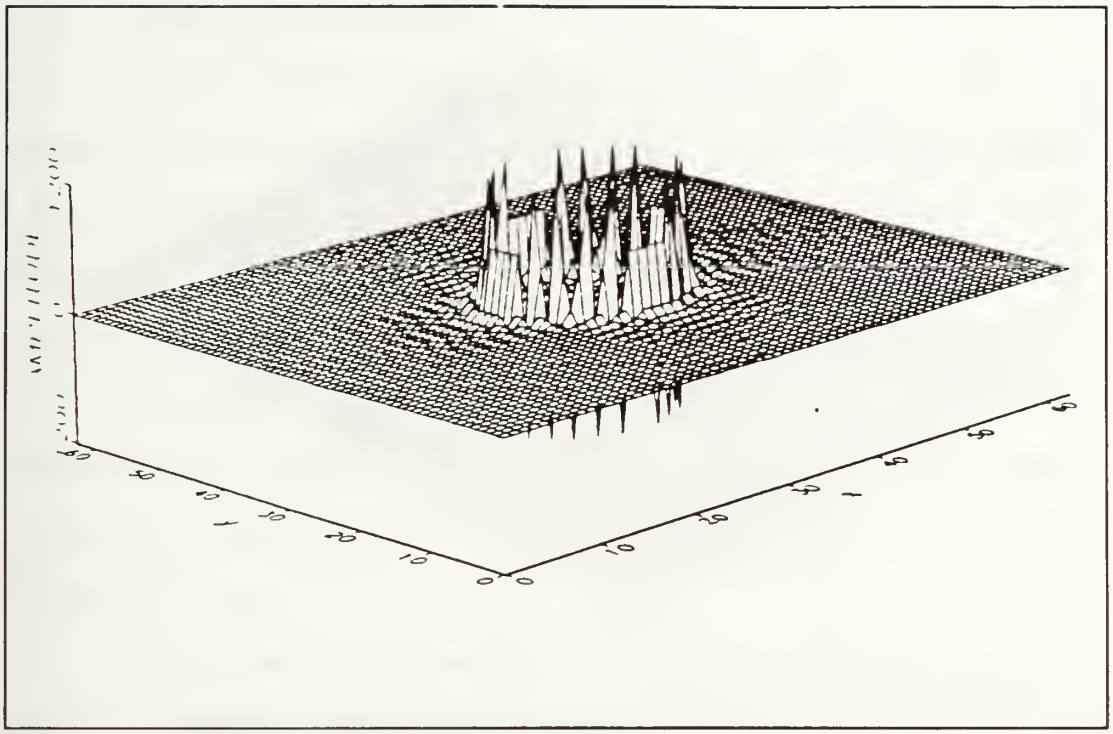

Figure 22. SHFT_OUTPUT2 (inverse transform of product of derivative filter and transformed input).

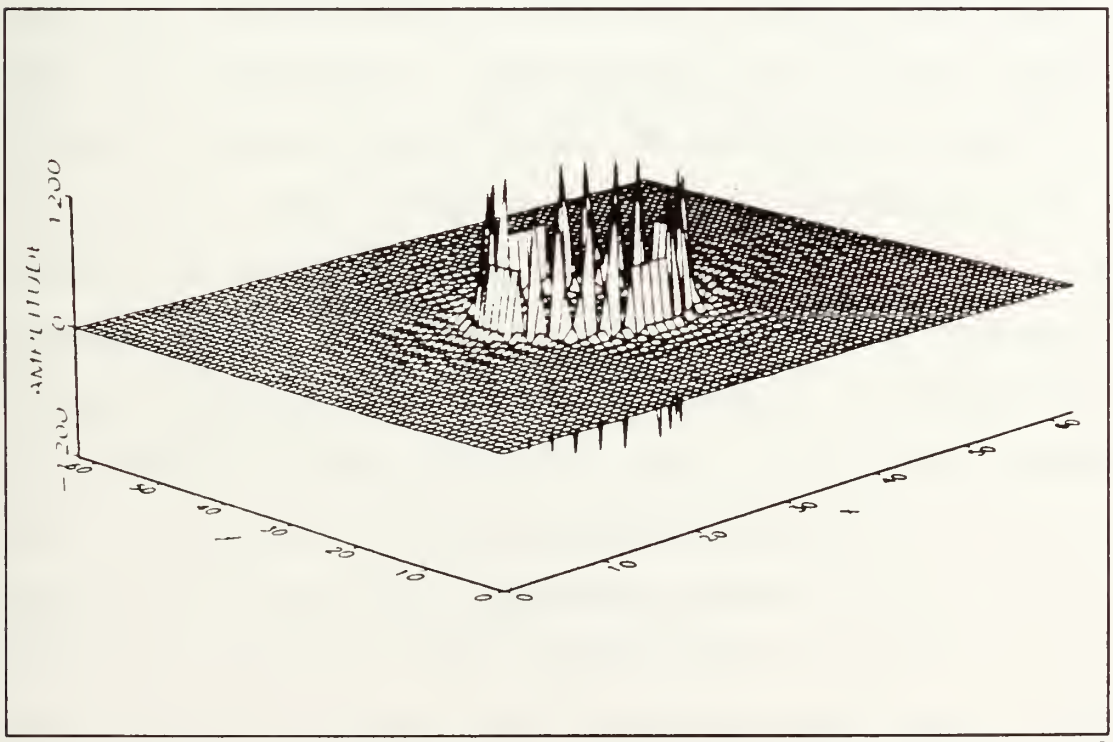

Figure 23. SHFT_OUTPUT (sum of SHFT_OUTPUT1 and SHFT_OUTPUT2).

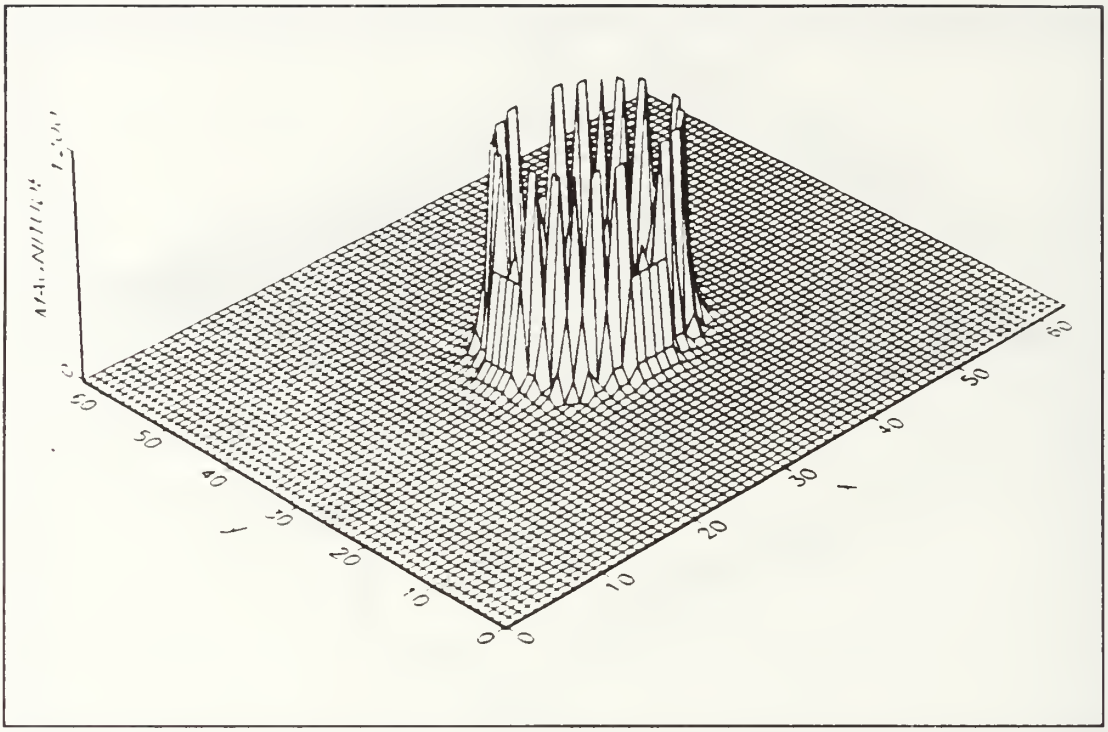

Figure 24. SHFT_OUTABS (magnitude of SHFT_OUTPUT).

3. Time incrementing

It is important to understand that the transient input function may be viewed as a grid of point sources, each of which is contributing equally to the output field. At any position on the observation plane, the output field is initially formed by the source point closest to that location. The field continues to be formed as time progresses by those source points successively more distant. Thus, the output is the sum of the effects of all the point sources over a finite period of time.

Examples of the Bessel filter have been shown in Figs. 10-13. Recall the description that the filter gives the appearance of collapsing on itself as time progresses. The derivative filter exhibits a similar tendancy over time, as Figs. 25-28 illustrate. Fshft_output1, which is the element-by-element product of the Bessel filter and the transform of the input function, becomes more narrow in the center, and the adjacent oscillations smooth out (refer to Figs. 29-32). This narrowing effect would lead one to believe that the spatial transform of Fshft_output1 should broaden, and that is what will be seen with shft_output1 in Figs. 33-36. Fshft_output2, which is the result of multiplying the derivative filter and transform of the input element-by-element, has an enormous magnitude (see Figs. 37-40). Shft_output2 initially is quite spiked (see Figs. 41-44), but begins to smooth out over time

and remains at least an order of magnitude larger than shft_output1. For this reason, the sum of the two, shft_output, is dominated by shft_output2 throughout. Figures 45-48 illustrate this dominance. Shft_outabs is the magnitude of shft_output (see Figs. 49-52).

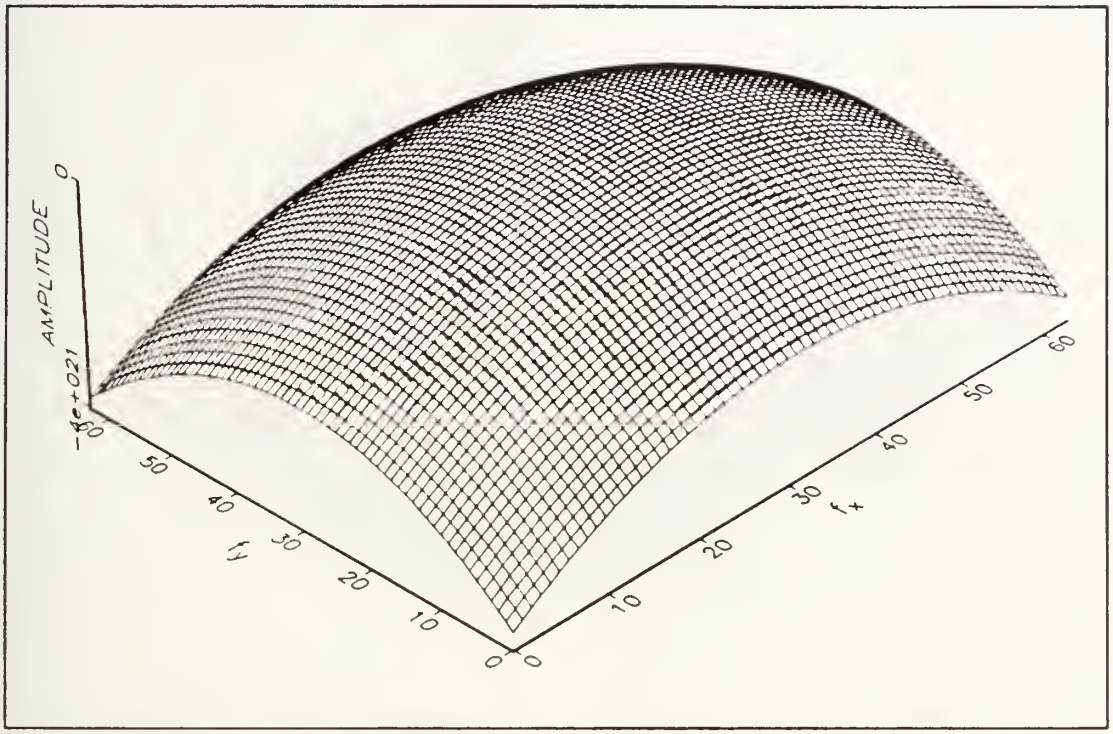

Figure 25. DER_FIL at time slice 1 (derivative filter).

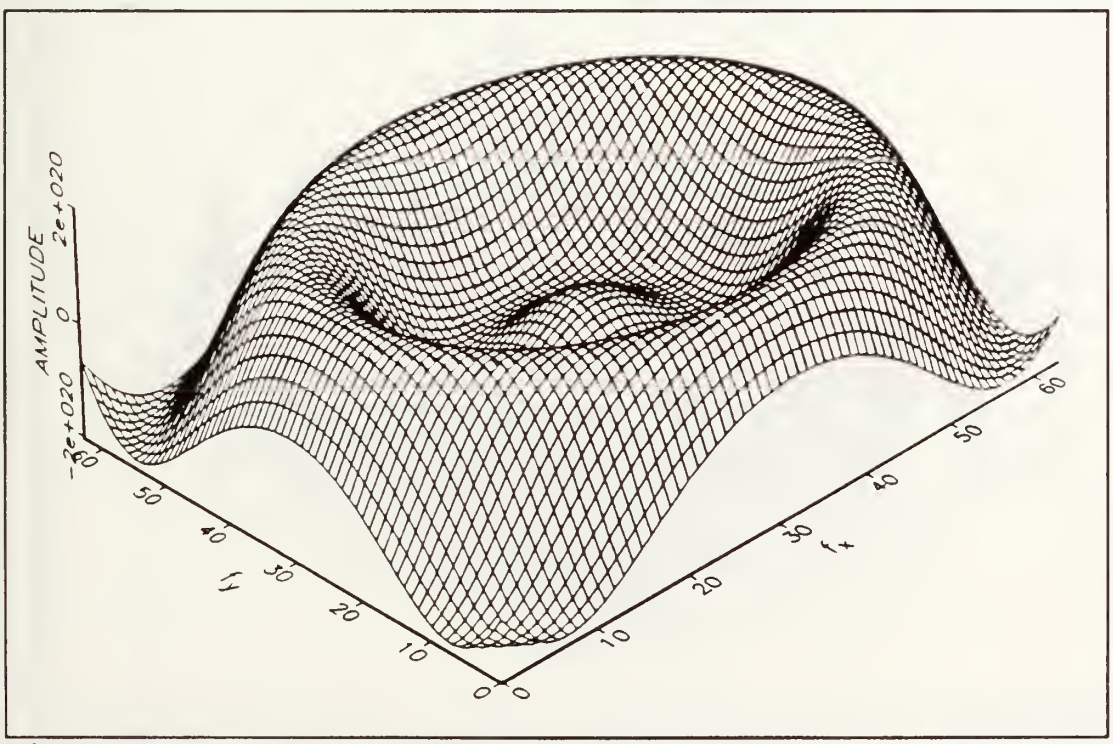

Figure 26. DER_FIL at time slice 3.

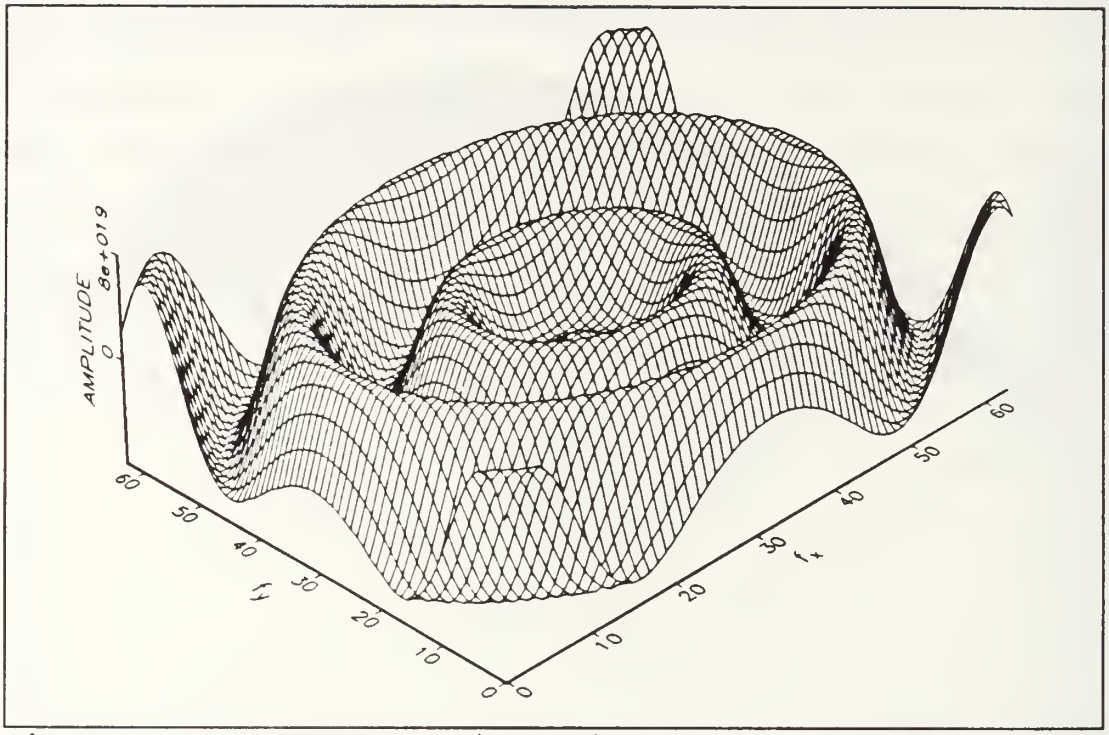

Figure 27. DER_FIL at time slice 8.

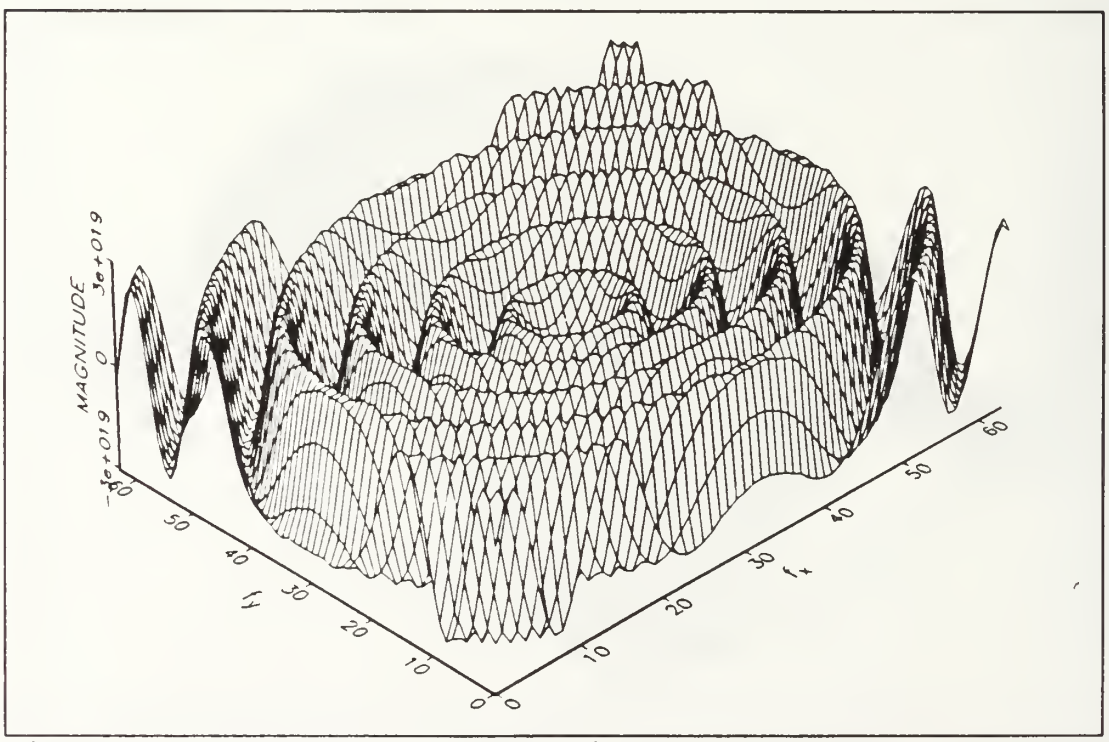

Figure 28. DER_FIL at time slice 23.

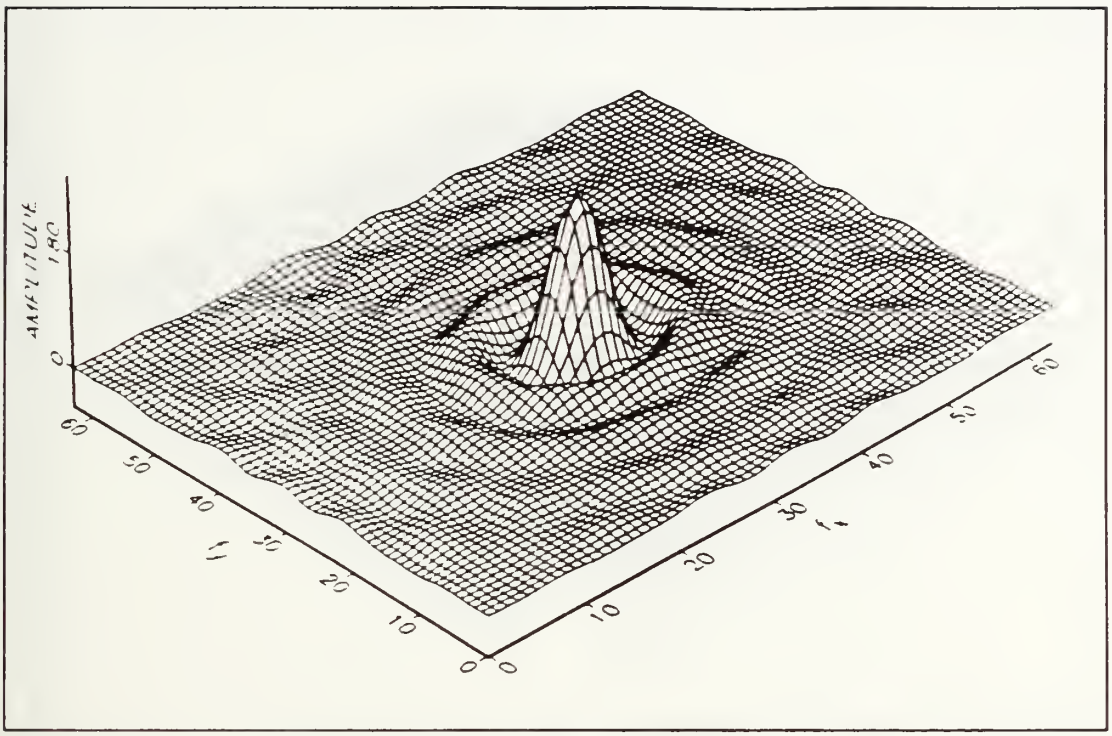

Figure 29. FSHFT_OUTPUT1 at time slice 1 (product of Bessel filter and transformed input).

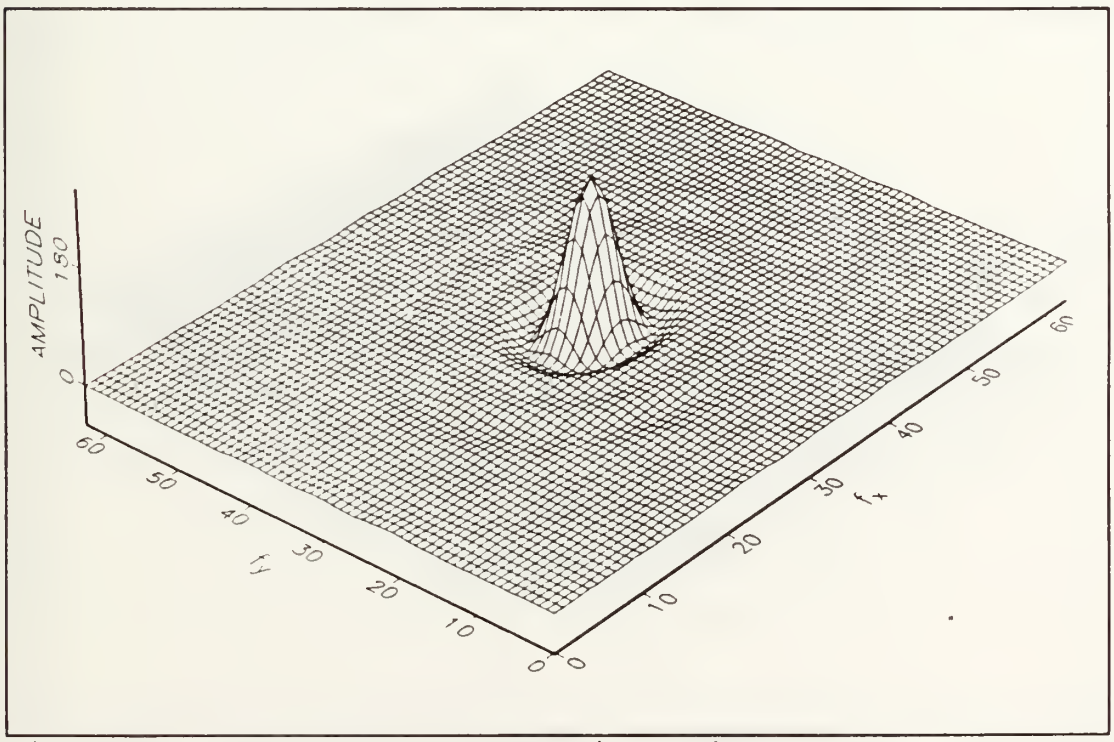

Figure 30. FSHFT_OUTPUT1 at time slice 3.

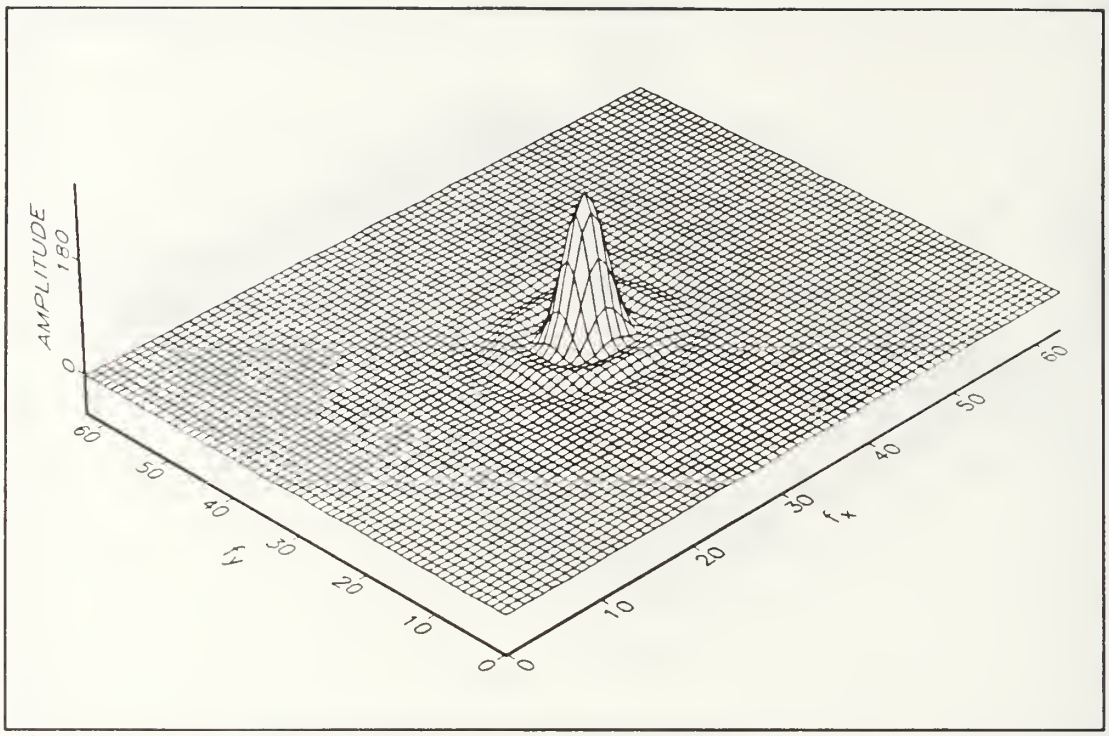

Figure 31. FSHFT_OUTPUT1 at time slice 8.

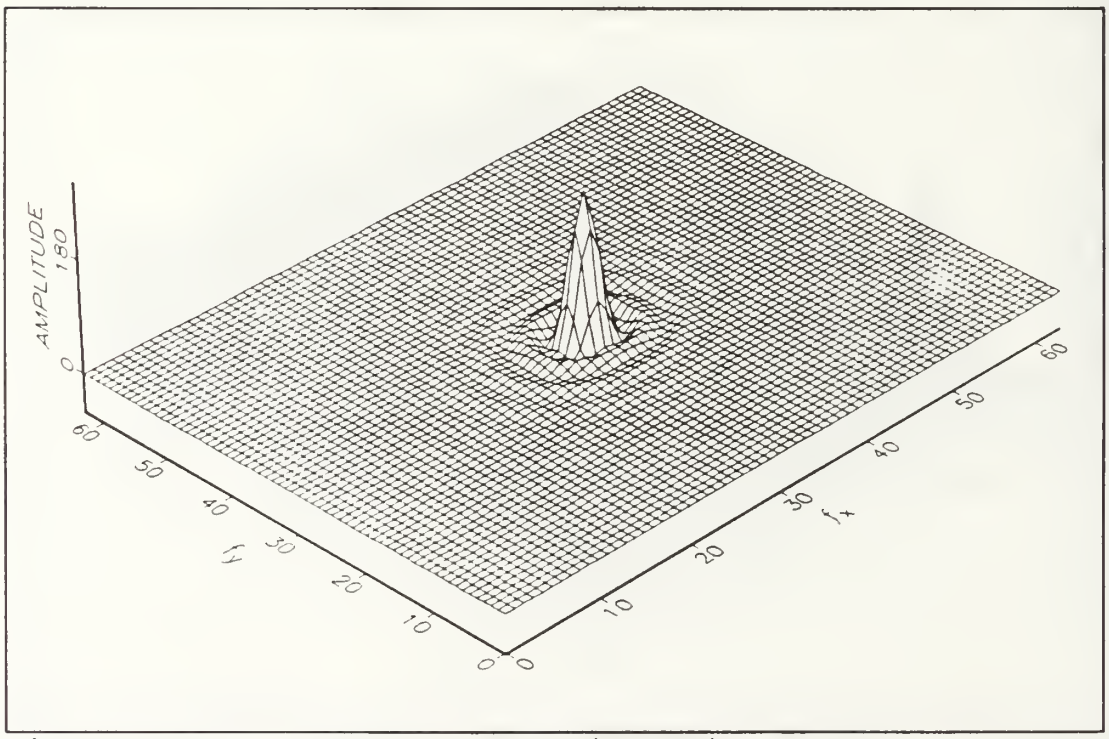

Figure 32. FSHFT_OUTPUT1 at time slice 23.

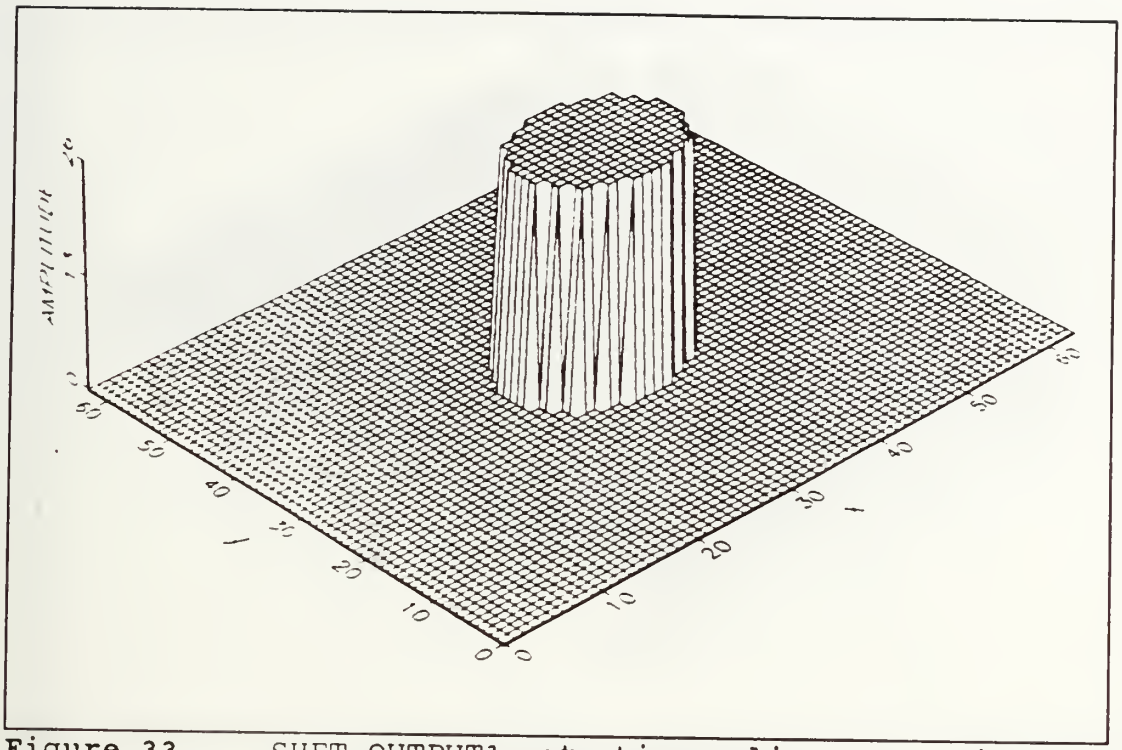

Figure 33. SHFT_OUTPUT1 at time slice 1 (inverse transform of product of Bessel filter and transformed input).

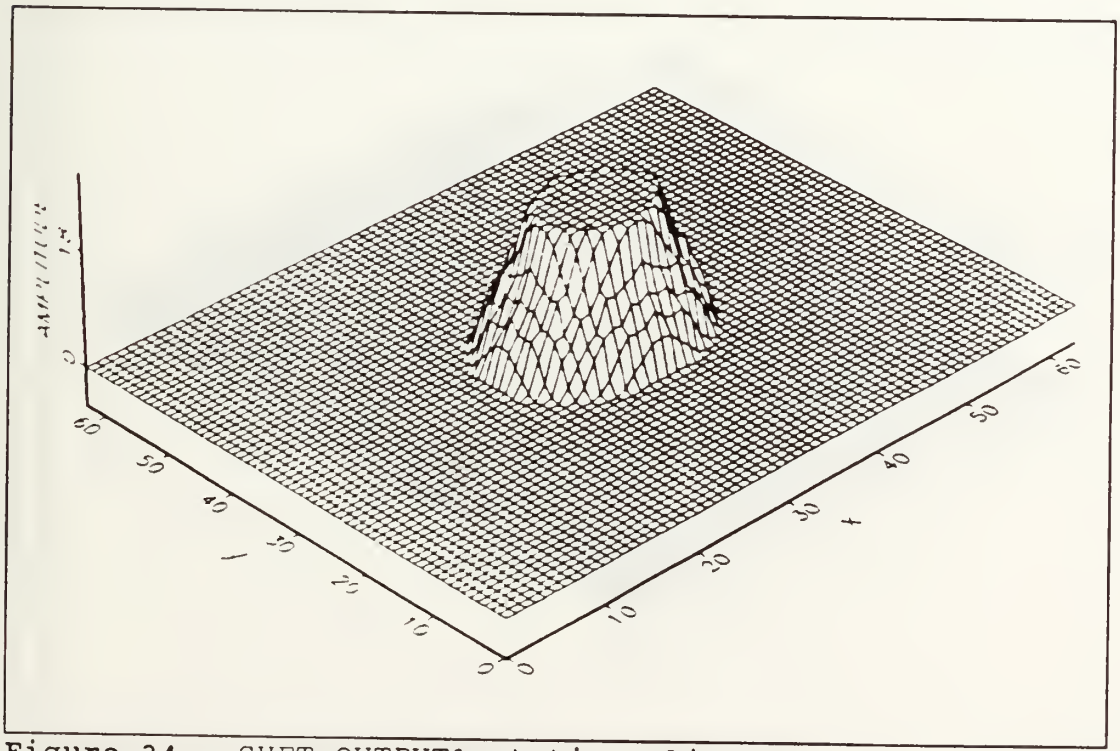

Figure 34. SHFT_OUTPUT1 at time slice 3.

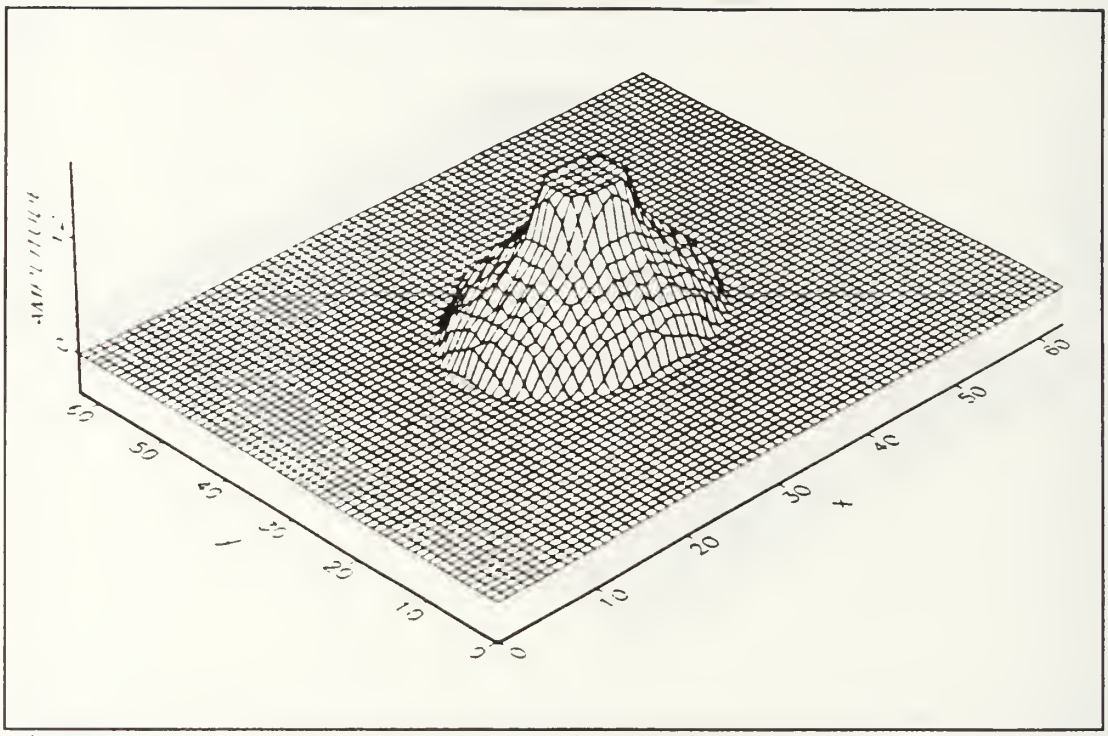

Figure 35. SHFT_OUTPUT1 at time slice 8.

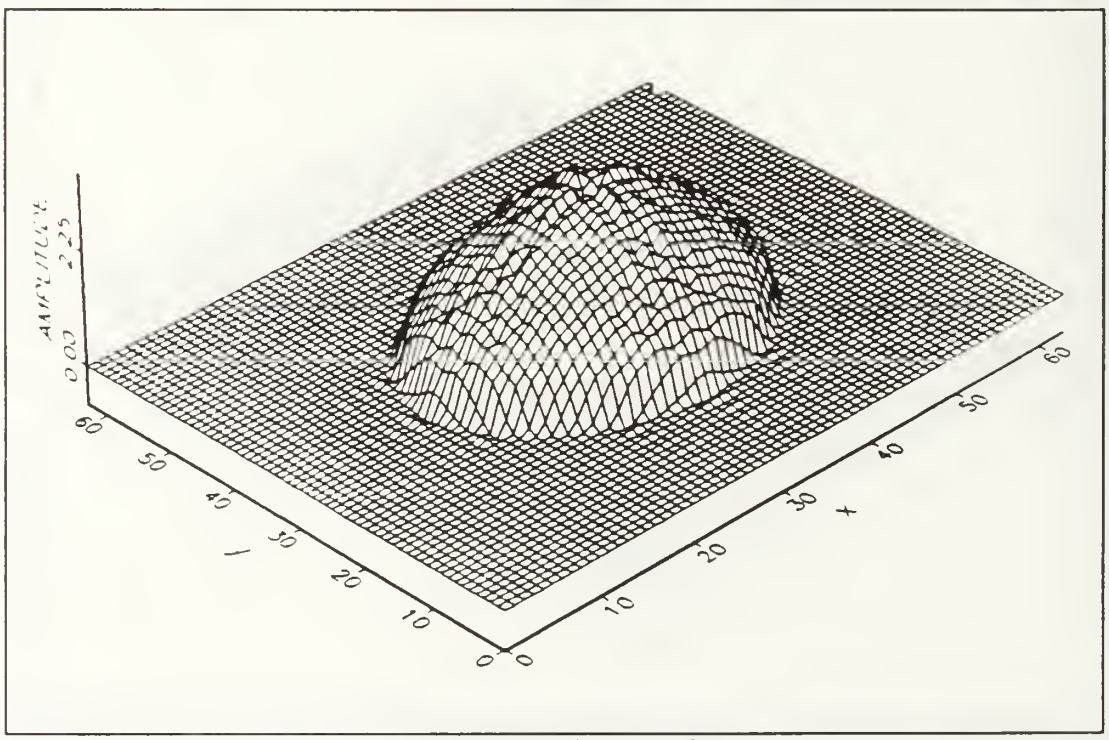

Figure 36. SHFT_OUTPUT1 at time slice 23.

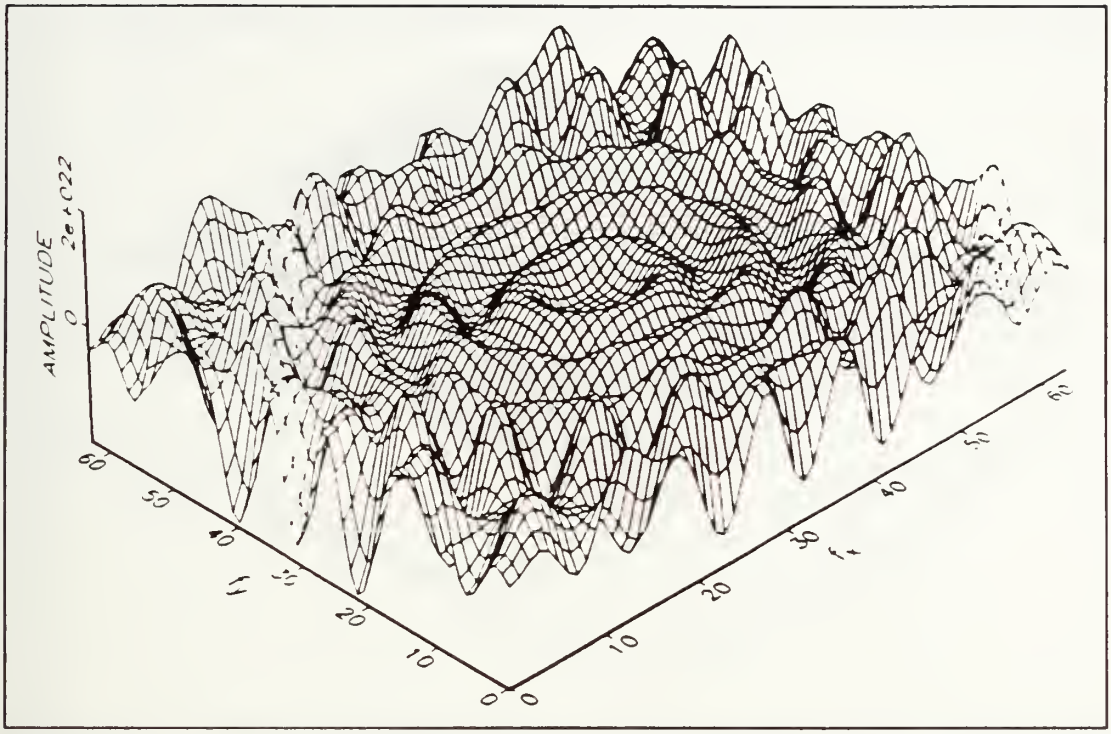

Figure 37. FSHFT_OUTPUT2 at time slice 1 (product of derivative filter and transformed input).

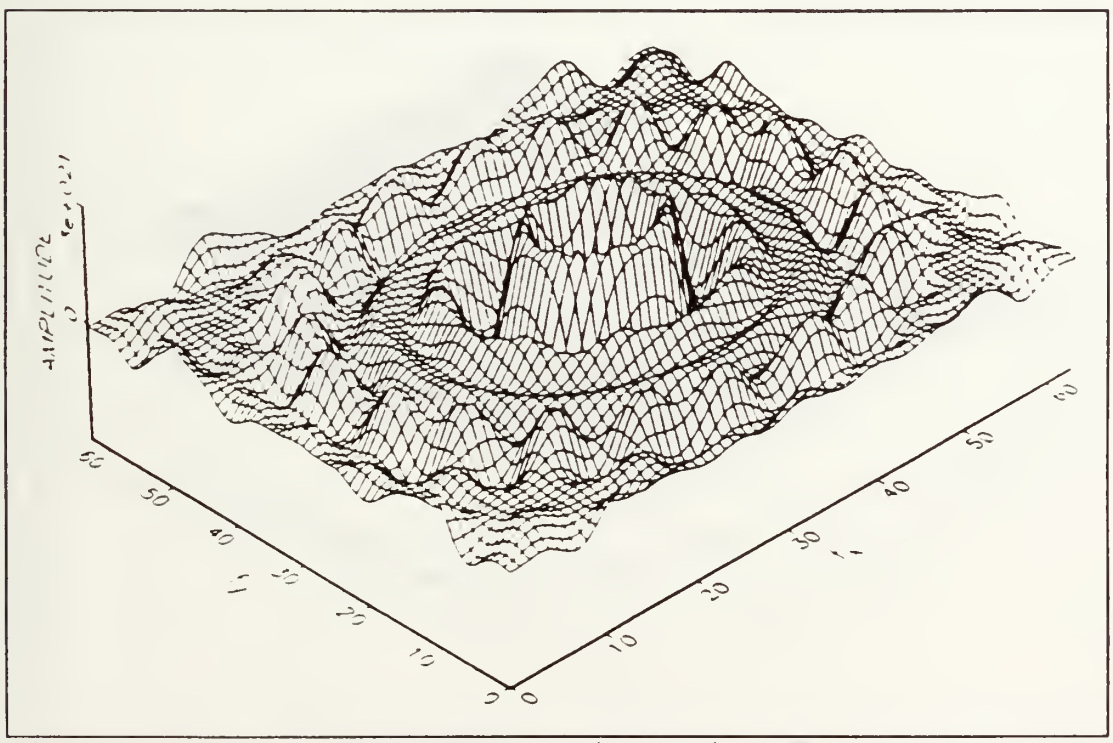

Figure 38. FSHFT_OUTPUT2 at time slice 3.

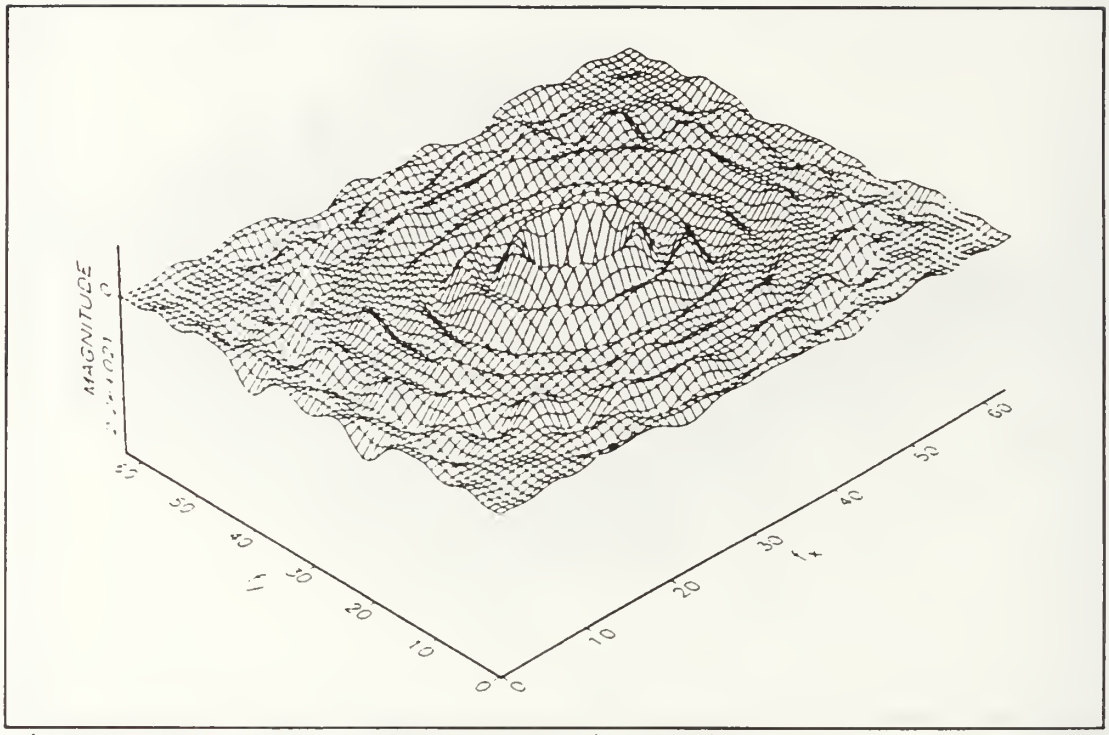

Figure 39. FSHFT_OUTPUT2 at time slice 8.

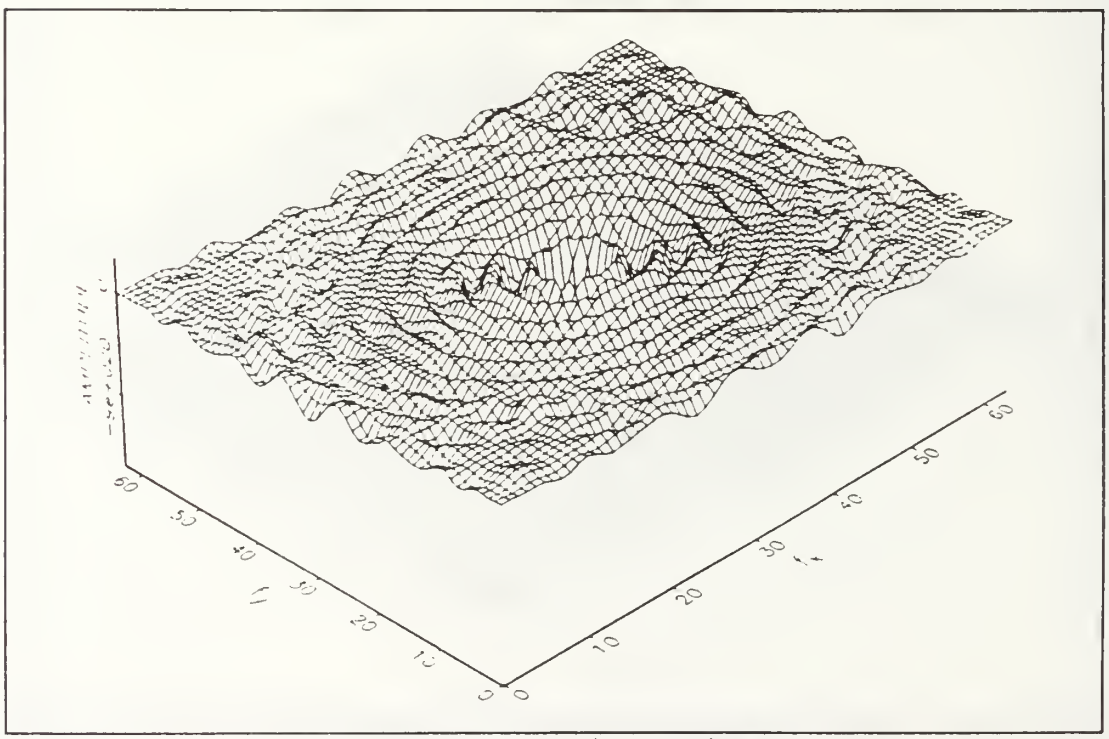

Figure 40. FSHFT_OUTPUT2 at time slice 23.

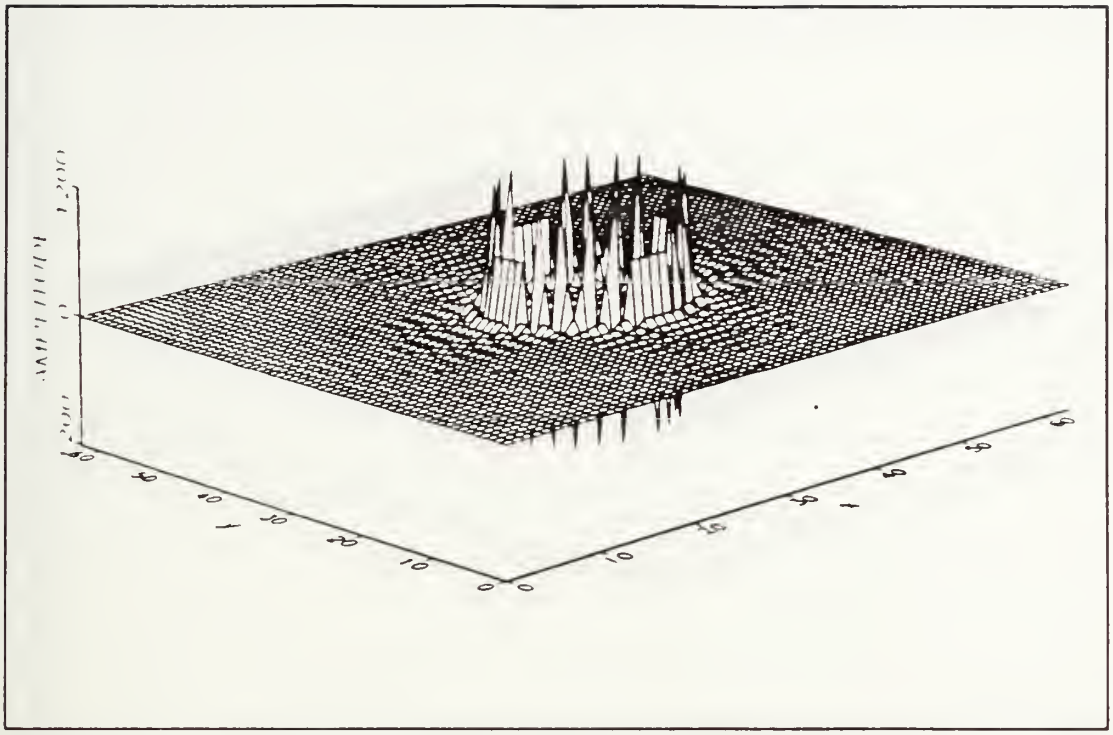

Figure 41. SHFT_OUTPUT2 at time slice 1 (inverse transform of product of derivative filter and transformed input).

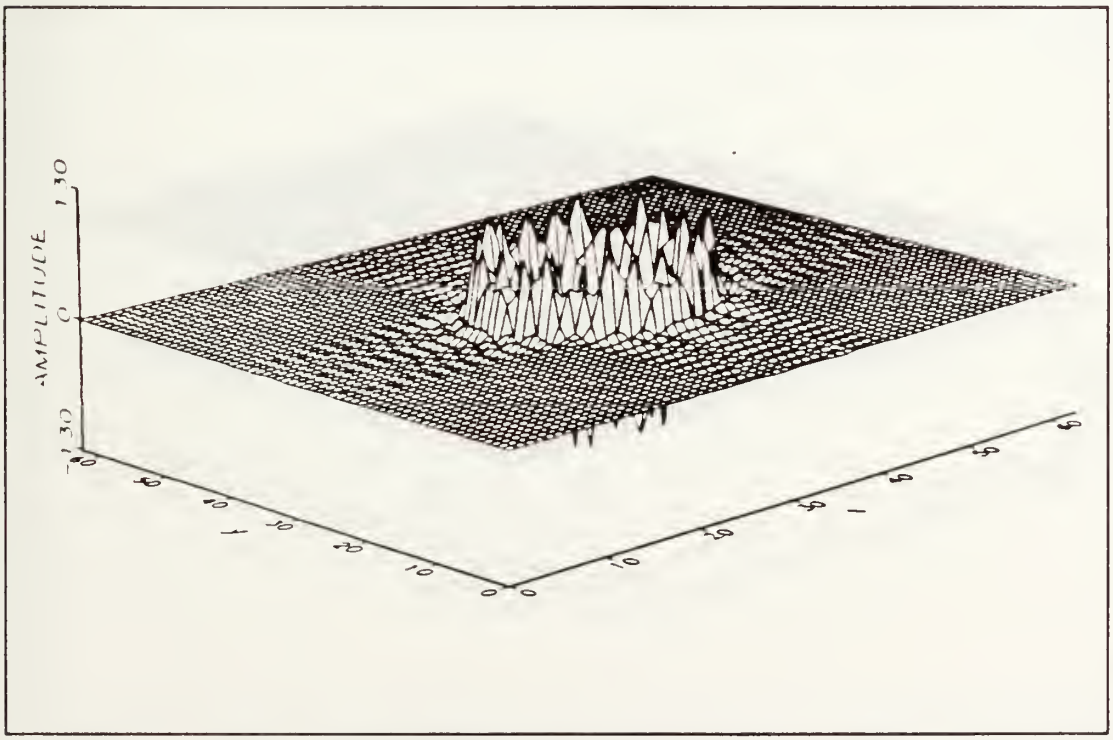

Figure 42. SHFT_OUTPUT2 at time slice 3.

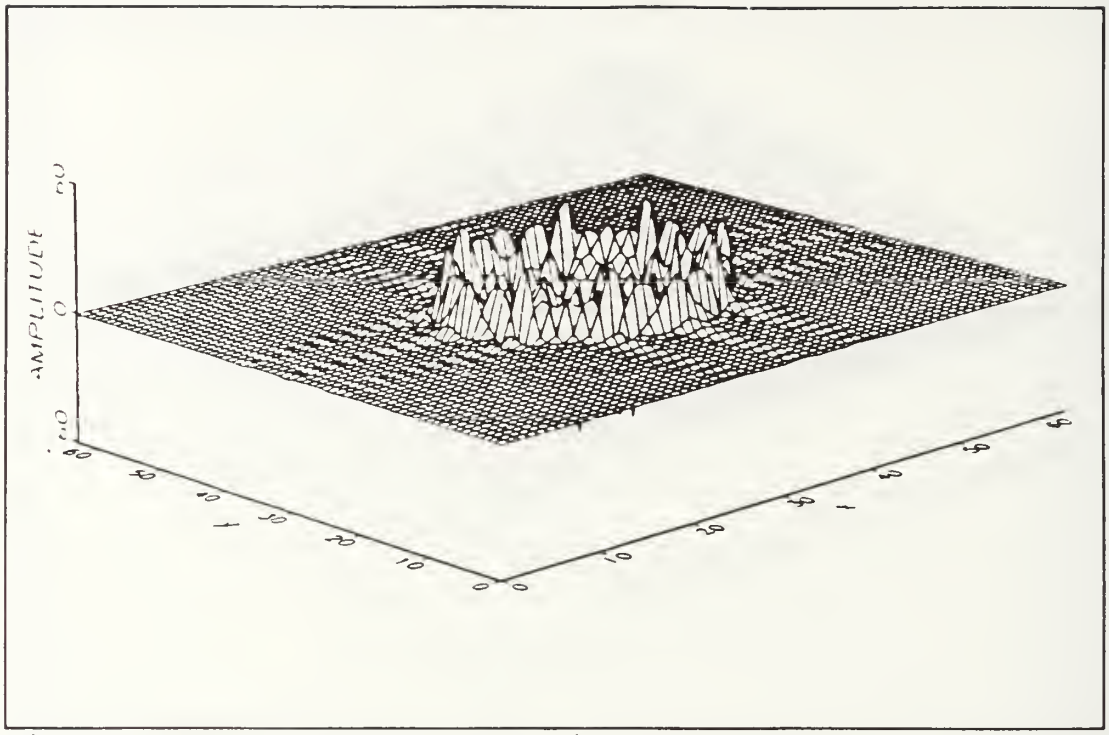

Figure 43. SHFT_OUTPUT2 at time slice 8.

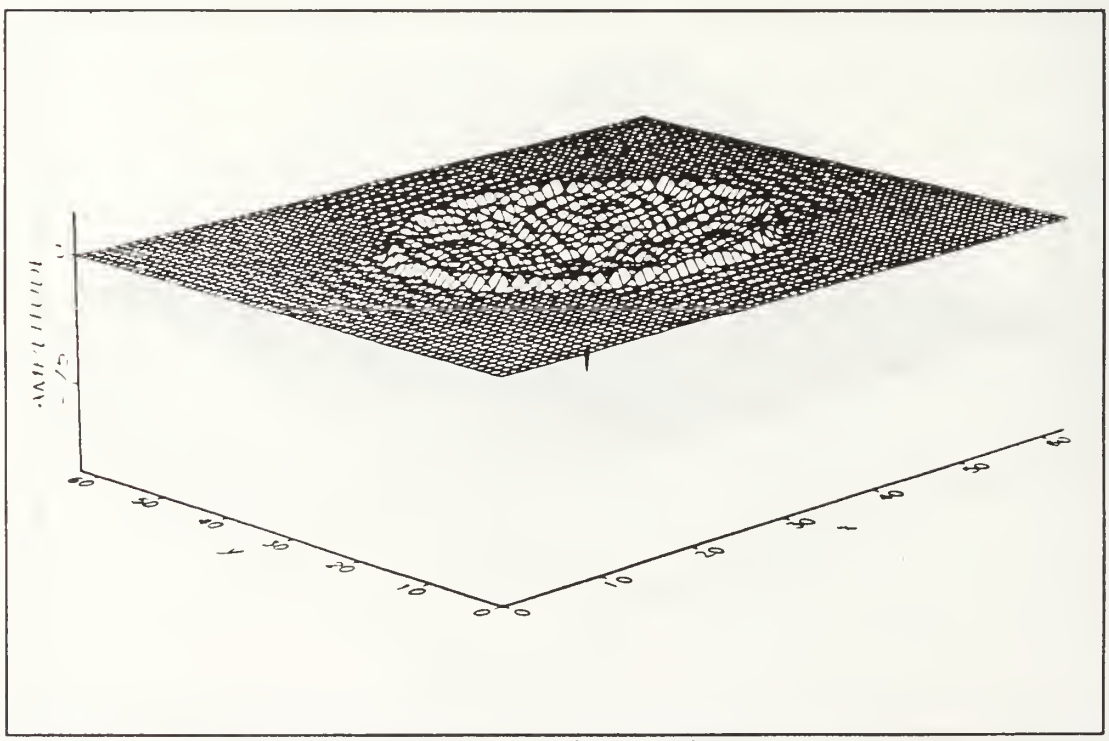

Figure 44. SHFT_OUTPUT2 at time slice 23.

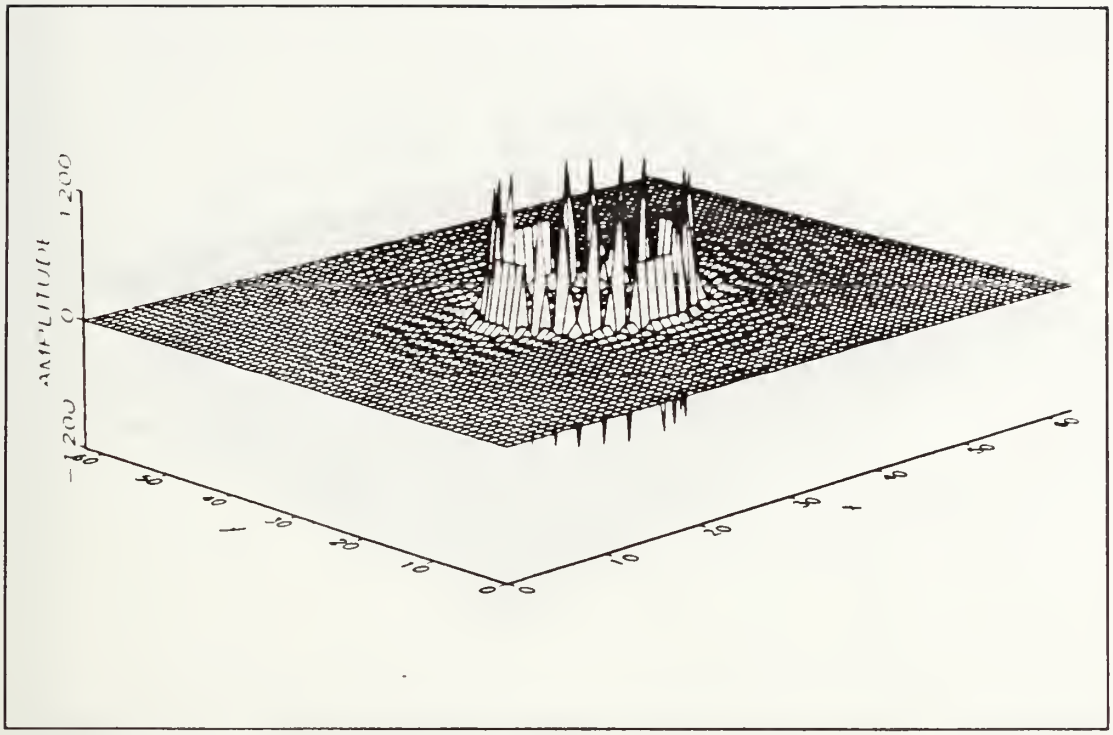

Figure 45. SHFT_OUTPUT at time slice 1 (sum of SHFT_OUTPUT1 and SHFT_OUTPUT2).

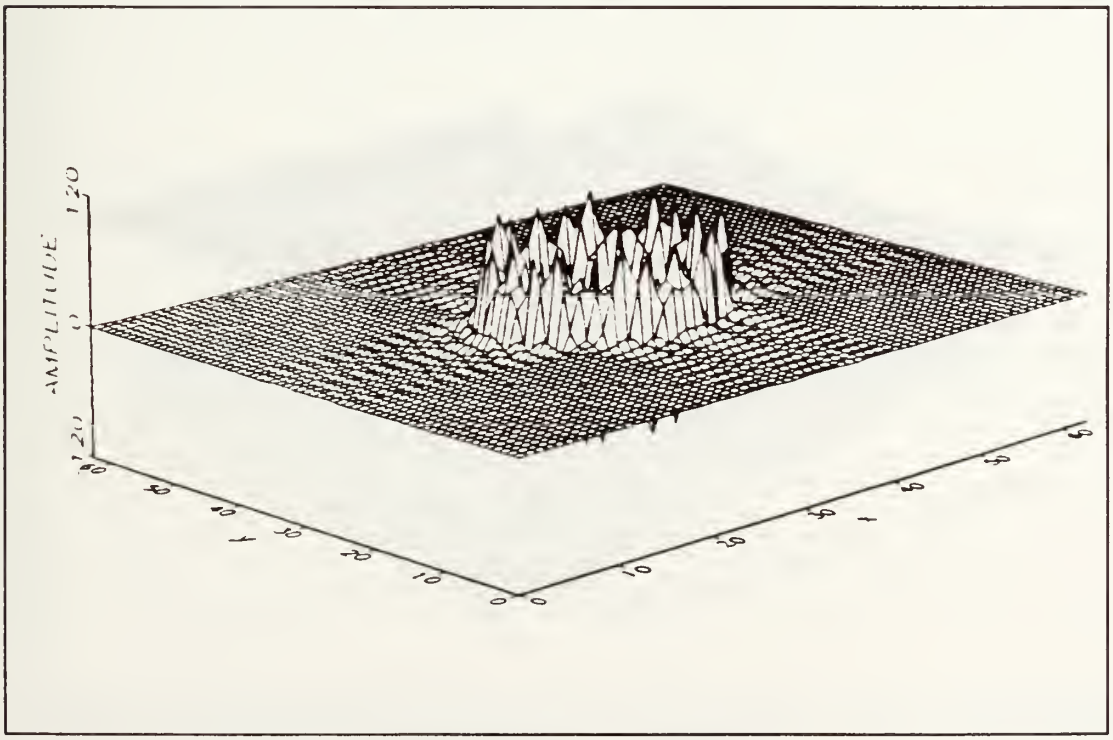

Figure 46. SHFT_OUTPUT at time slice 3.

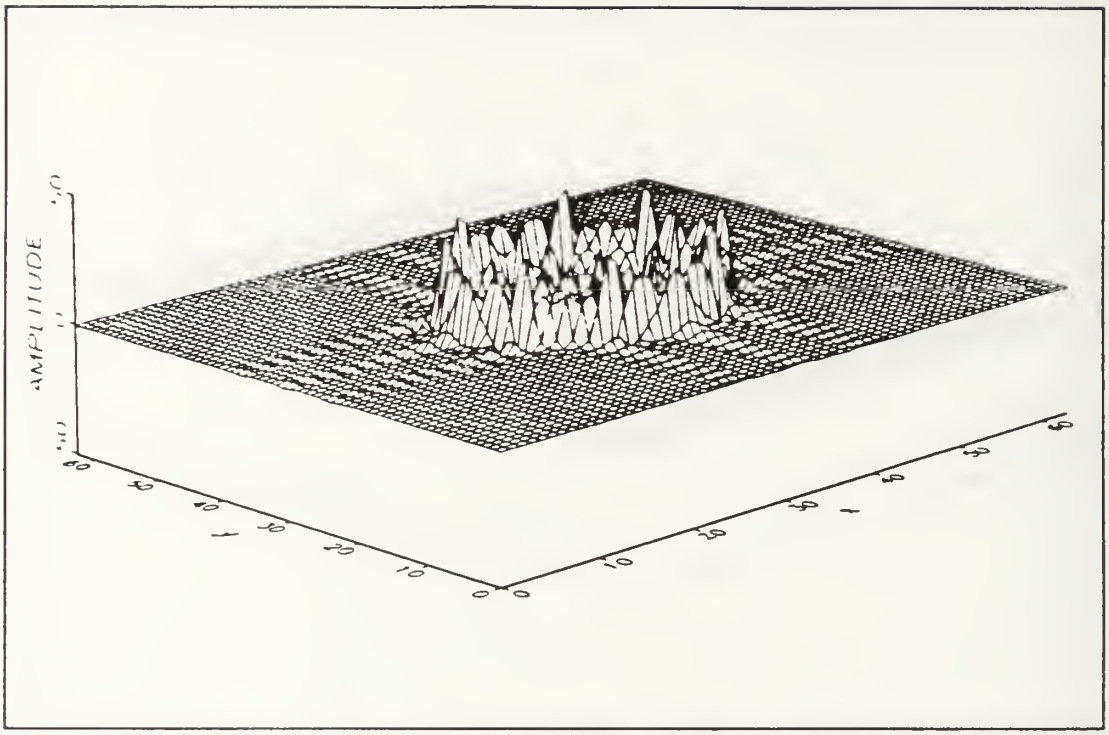

Figure 47. SHFT_OUTPUT at time slice 8.

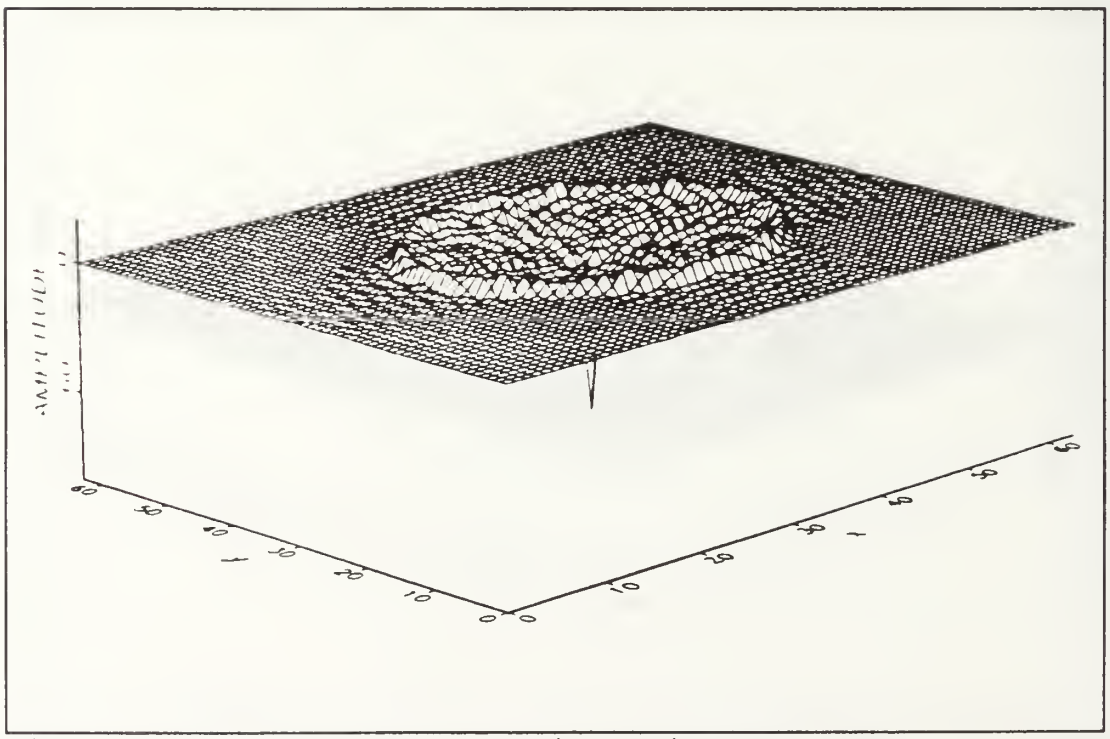

Figure 48. SHFT_OUTPUT at time slice 23.

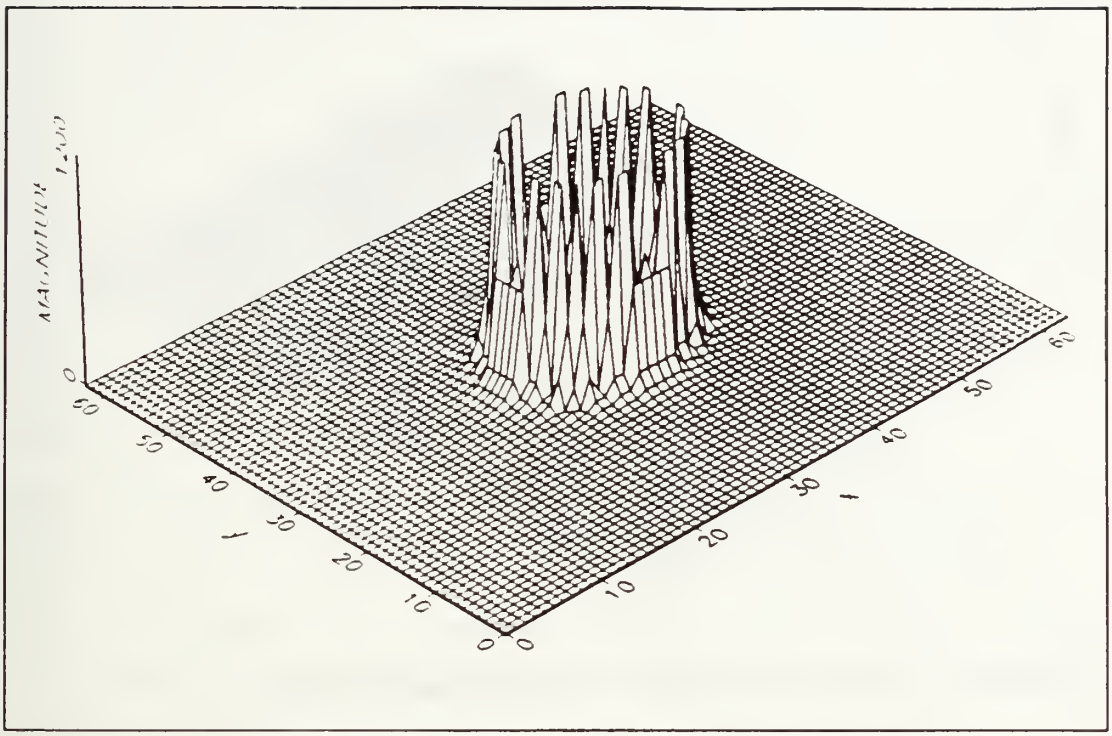

Figure 49. SHFT_OUTABS at time slice 1 (magnitude of SHFT_OUTPUT).

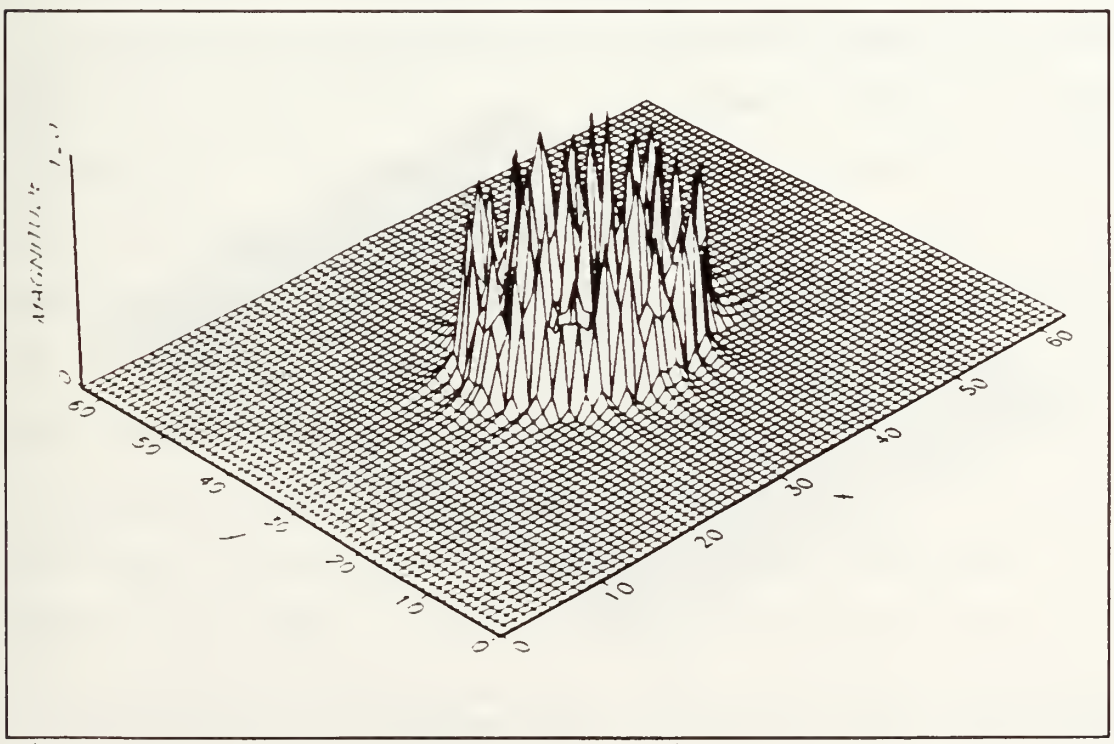

Figure 50. SHFT_OUTABS at time slice 3.

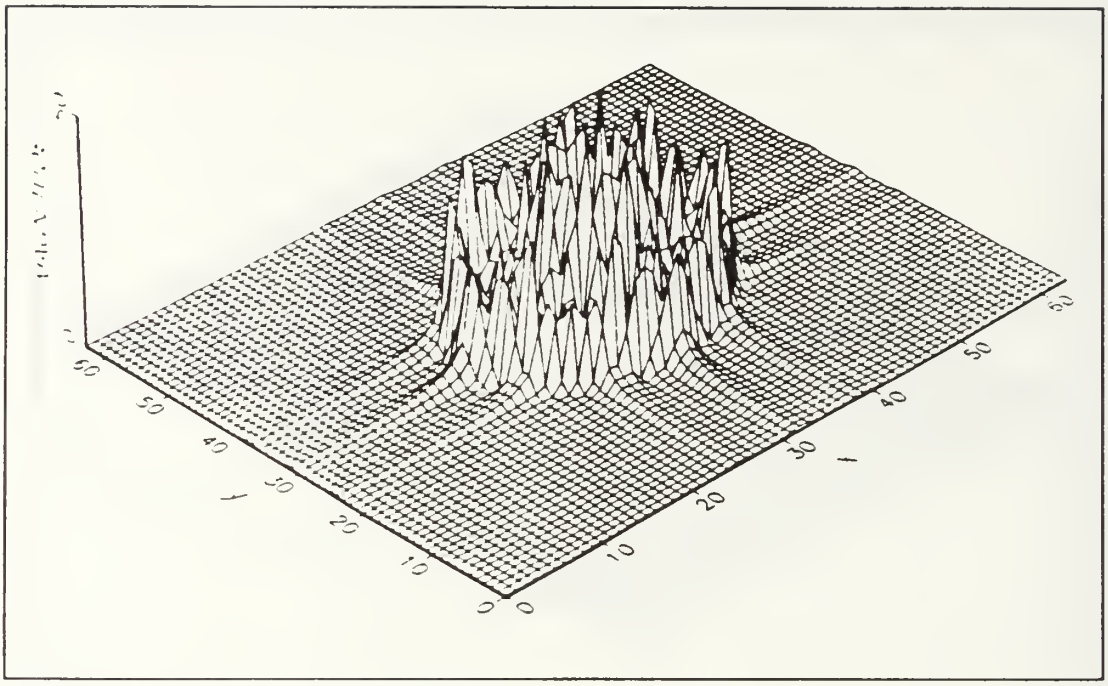

Figure 51. SHFT_OUTABS at time slice 8.

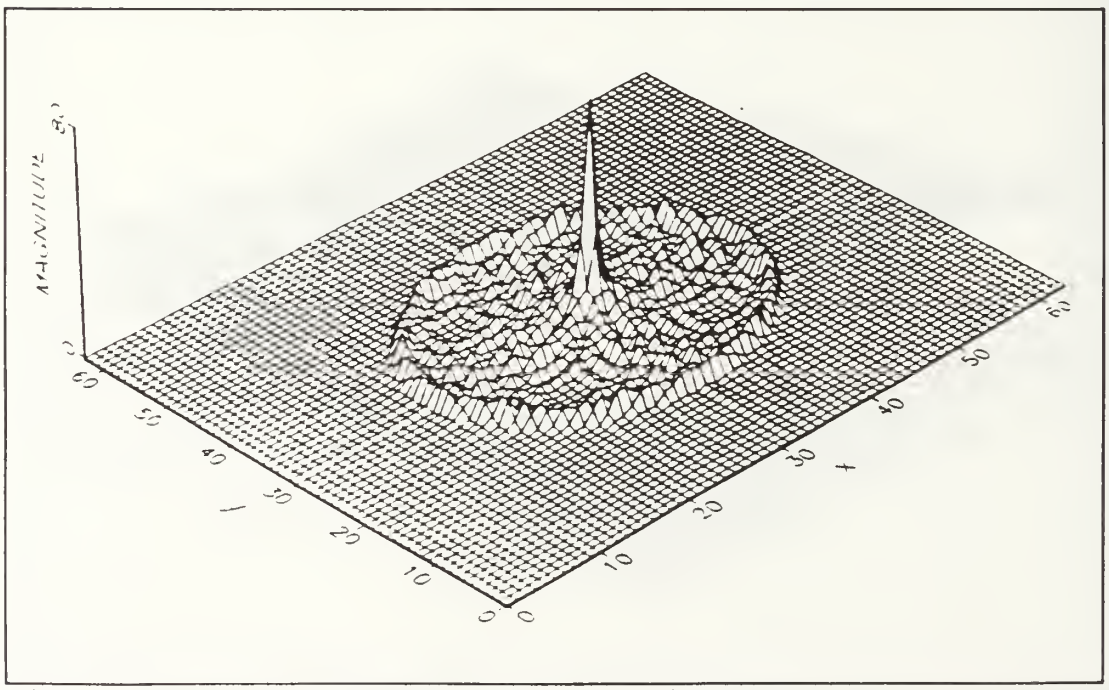

Figure 52. SHFT_OUTABS at time slice 23.

IV. NUMERICAL SIMULATION

This chapter addressess the output of the program for both a circular and square excitation function, given a particular set of defining parameters. The parameters used in the simulation are explained in the first section. A brief discussion of the execution time follows. The third section presents some information about the graphics program AXUM which was used to construct nearly all of the figures presented throughout the thesis. Finally, the optical field predictions for each excitation function are shown.

A. EXPLANATION OF DEFINING PARAMETERS

The defining parameters are those variables found at the beginning of the OPTFIL code which either OPTFIL or OPTPROP utilize for the basic problem setup. Examples of the parameters include N, M, Step, time_max, etc.; each has been addressed functionally in previous chapters. This discussion will cite the specific values used in the numerical simulation which produced the results shown at the end of this chapter.

The parameter N defines the number of points in the square matrix and the value of N=64 was used for this case. As N defines the number of data points, a 128x128 matrix would provide greater resolution at a cost of longer computation

times. Further research to investigate whether the resolution is warranted would be useful. The center of the matrix, N0, was 33. Step, the number of time slices prior to z/c for which a zero output would be recorded, was set at 3. A value of 64 was selected for M which resulted in 61 time slices (M-Step). The distance between the source plane and the observation plane, z, was set to 80 millimeters. This value was determined by scaling the value of z used in prior acoustic simulations by the ratio of propagation speeds. Time_max was determined in a similar fashion, although a bit of trial and error was also necessary. Its value in this thesis was 0.95 nanoseconds. Rho, also based on acoustic work, was 200 spatial frequency units, and c, the speed of light, was set to 3E8 meters per second. T_eps is the smallest time value between successive Bessel filters for which a measurable difference could be obtained. Anything smaller meant that the subtraction of the two resulted in a zero output. By such reasoning, t_eps was set to 23.5E-25 seconds. Table 1 summarizes the defining parameters and the repective values used in this simulation.

B. EXECUTION TIME

Although no formal goal was set as far a desired maximum execution time is concerned, it is naturally of interest. The sequence of OPTFIL and OPTPROP for a 64x64 matrix and 61 time slices was run both on an Intel 386/20 MHz machine with a math

Table I. VALUES OF DEFINING PARAMETERS.

PARAMETER	VALUE	DEFINITION
N	64	size of the square matrix
M	64	total number of time slices
z	80 mm	distance from source plane to observation plane
time_max	0.95 ns	maximum observation time
rho	200	spatial radius
c	3e8 m/s	velocity of propagation
t_eps	23.5e-25s	difference equation time value
Step	3	time increments prior to z/c

co-processor and on an Intel 486/33 MHz. The 386/20 required slightly over forty minutes to generate all the Bessel filters and ten minutes to complete OPTPROP to the final output. The 486/33 ran approximately five times faster needing slightly less than eight minutes for OPTFIL and a bit over 2 minutes for OPTPROP.

C. OVERVIEW OF AXUM

An alternative graphics program was required to produce 3-D plots with scaled vertical axes, since MATLAB 3-D mesh displays do not incorporate that feature.

AXUM is an advanced technical graphics and data analysis program produced by TriMetrix, Inc. It requires a minimum of 512K of memory and generates 3-D contour plots in a relatively

straightforward fashion [Ref. 11]. Data is imported into AXUM by first storing the MATLAB data in ASCII format from OPTPROP. This was easily done, and there are several examples of the coding steps necessary included in the source code in Appendix F.

Once the machinations of data manipulation, axes definition, scaling and labeling, et al. were accomplished, a truly useful feature was found in a menu entitled "Edit Graph". It allowed interactive modifications to be done which includes changing the perspective either horizontally or vertically. This capability was used to orient the SHFT_OUTPUT illustrations of the previous chapter by rotating the axes vertically.

D. EXAMPLES

The two source shapes to be presented are the uniform circular excitation and the uniform square excitation. Figure 53 illustrates the circular case with a diameter of 17 shown previously in Chapter 3. Figure 54 shows the calculated spatial impulse response, given the defining parameters cited in Section A above.

Recall that this displays the time progression of h(0,y,z,t) only, for each matrix generated per time slice. As expected, it is nearly spatially symmetrical, and, over time, the strength fades to zero. "Noise" is present between the outboard "tails" and the center of the response. Added

resolution may smooth that, in addition to the other jagged edges. Note also the decided increase in amplitude as the two inboard tails cross. This overall shape compares favorably with that found by Guyomar [Ref. 4].

The uniform square excitation is shown in Fig. 55. This particular one has a width of 25 on a 64x64 base. The output (see Fig. 56) is smoother than that of the circular piston with less signal variations seen throughout. The magnitude of the crossing middle tails is much less pronounced, yet, there is an odd increase in magnitude in the vicinity of the crossing. The fact that the crossing occurs at a later point in time can be explained by the wider excitation function. Again, the field strength tends to zero as time progresses.

This chapter has presented the optical field strength prediction for a particular set of defining parameters, using the uniform circular and uniform square excitation functions of specified spatial width. The width of either excitation function can be easily changed to study the associated effect on the field. In addition, two other excitation functions, with circular Gaussian and circular Bessel spatial distributions, are available. The resultant field strength predictions using these latter two are left for future investigation.

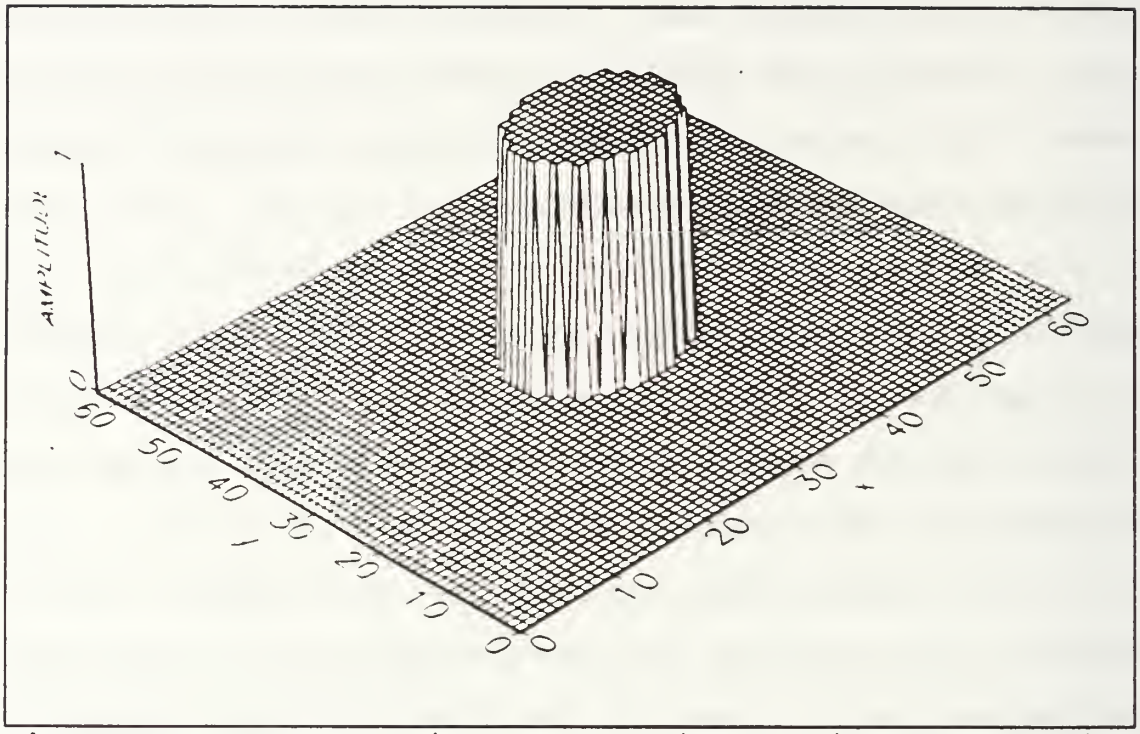

Figure 53. Input field for uniform circular spatial excitation.

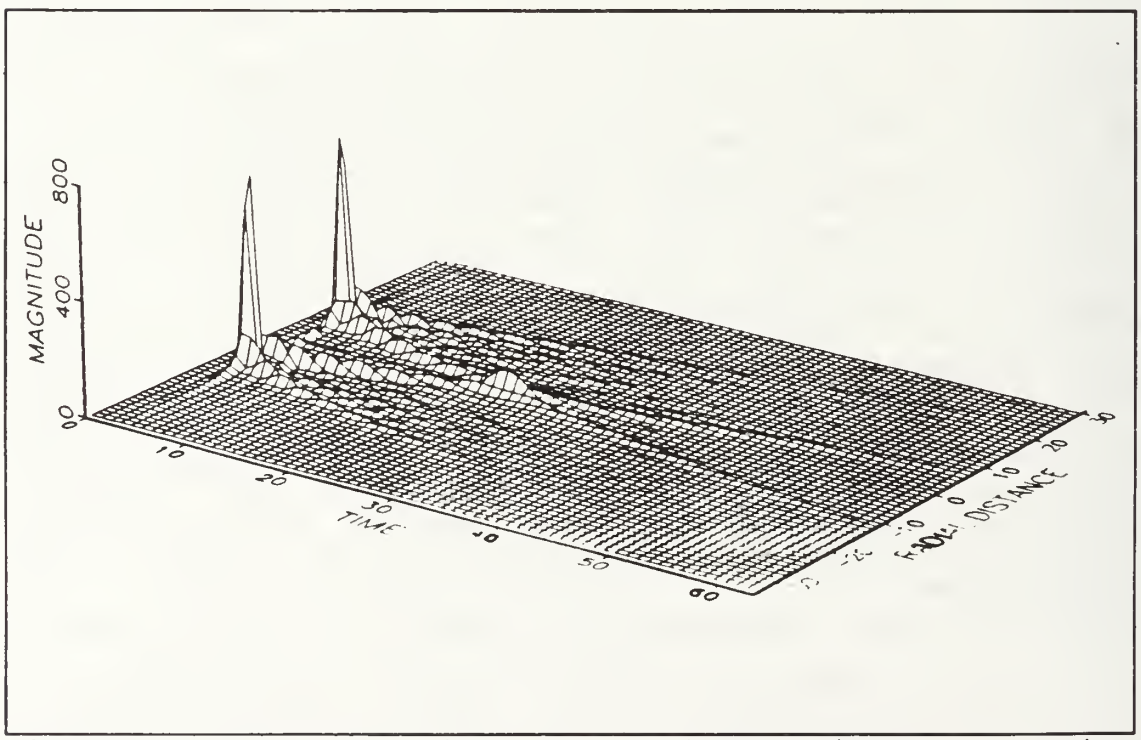

Figure 54. Output field for uniform circular spatial excitation.

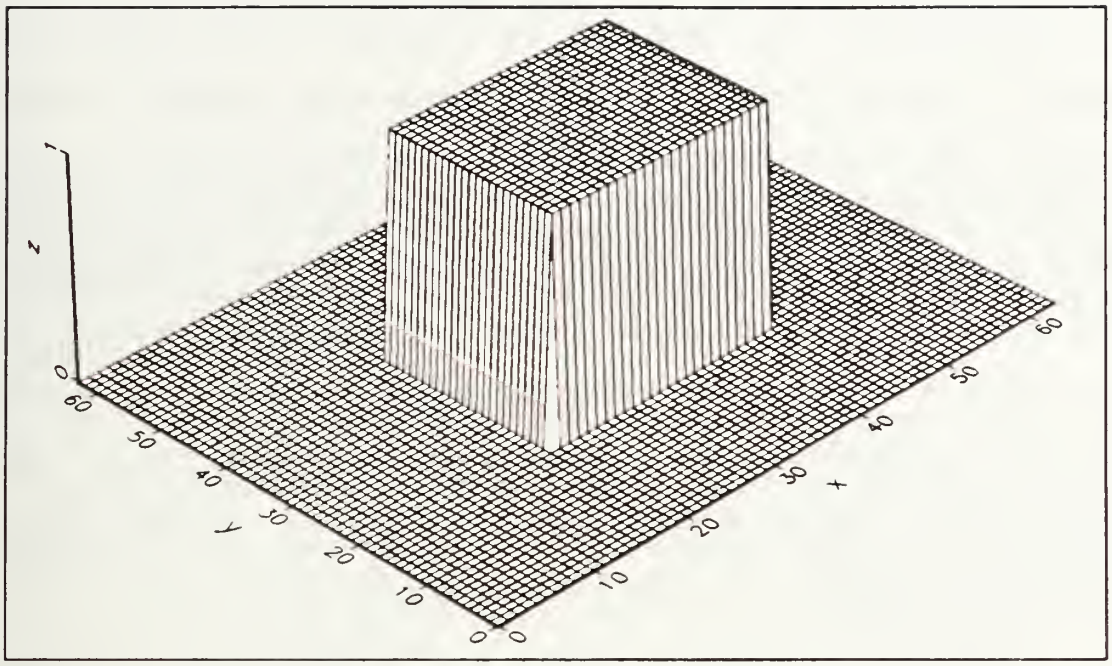

Figure 55. Input field for uniform square spatial excitation.

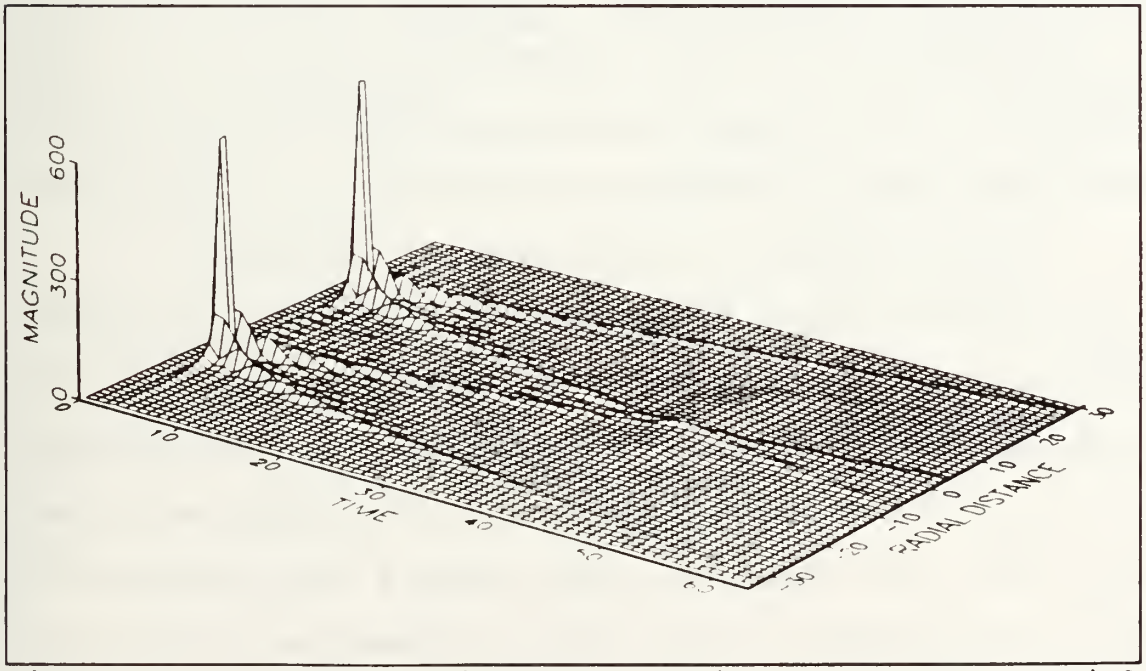

Figure 56. Output field for uniform square spatial excitation.

V. SUMMARY

This thesis investigated the possibility of solving complex simulations of scalar optical wave propagation utilizing the MATLAB program. Such a program was written in two sections, OPTFIL and OPTPROP, modularized in the sense that the more time-consuming process runs independently of the other. The results of the OPTFIL are stored to disk to be recalled by OPTPROP, as OPTPROP begins to run. A change in the excitation function, given a particular propagation geometry, does not require running both segments of the program but only the less time-intensive of the two (OPTPROP). This allows for a more rapid evaluation of the field distribution for numerous trials. Although no particular goal for a maximum run time was established, the results were satisfactory. This was particularly so on an Intel 486/33 MHz machine which ran both segments of the program in ten minutes and the OPTPROP alone in slightly over two minutes.

The systems theory supporting the concept of a spatial impulse response was presented, along with the mathematical derivation of the solution appropriate for an optical application. A functional discussion of the program was given in the body of the thesis for those readers whose interest is not in the specifics of the code. Appendices C and D do contain such specifics. Many figures have been incorporated

to augment the discussion of the program's operation. The spatial impulse responses for both a circular and a square uniform spatial excitation were computed. The outputs agreed favorably with previously computed results [Ref. 4]. These responses have been plotted in both 2-D and 3-D and a brief analysis presented.

Future investigation is open in several areas. The size of the base matrix should be increased to 128x128 to provide for additional resolution. A "two-sided" derivative approach to the difference equation may smooth the results somewhat. Work thus far has been limited to a square aperture but results with a circular base could prove of interest. A true understanding of the propagation will not be achieved until a means to show the results from the entire field, not just the center portion, is developed. Achieving this, detailed data analysis can progress.

APPENDIX A. EXAMPLES OF BESSEL FILTERS

This appendix contains additional examples of Bessel filters. The value of time at which each has been calculated is found in the figure caption.

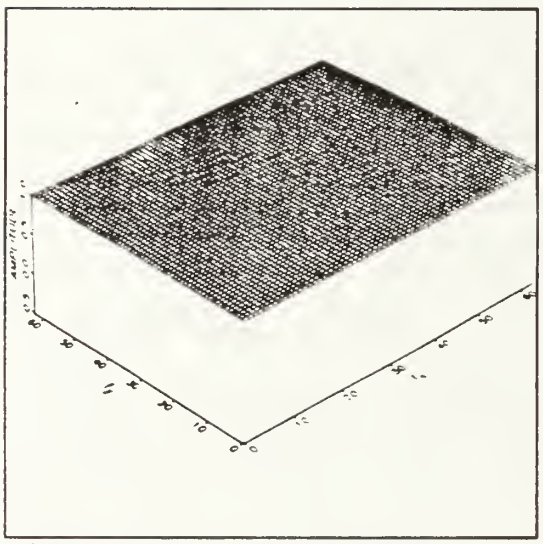

Figure 57. Time slice 1.

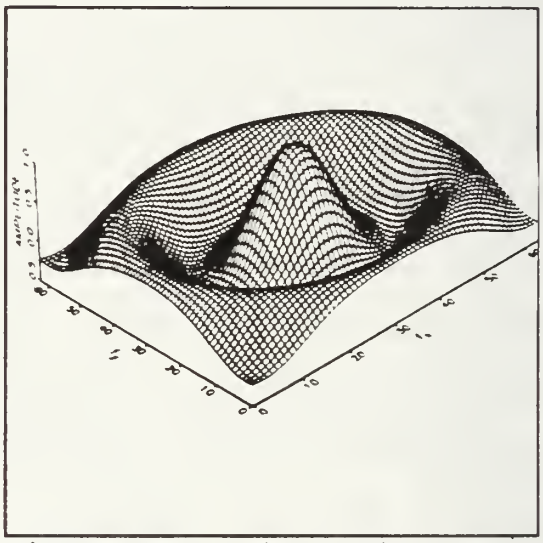

Figure 58. Time slice 3.

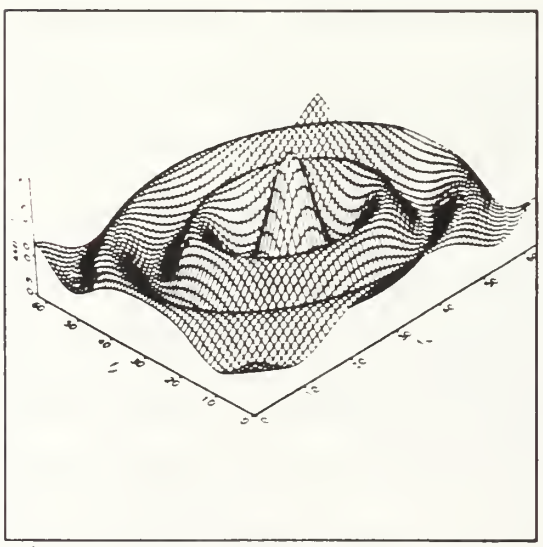

Figure 59. Time slice 8

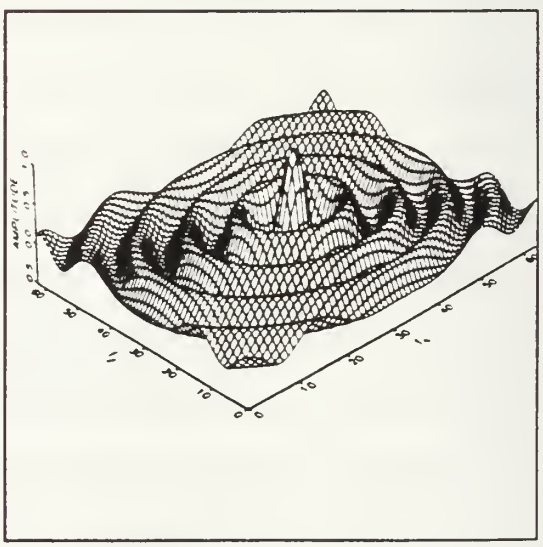

Figure 60. Time slice 18.

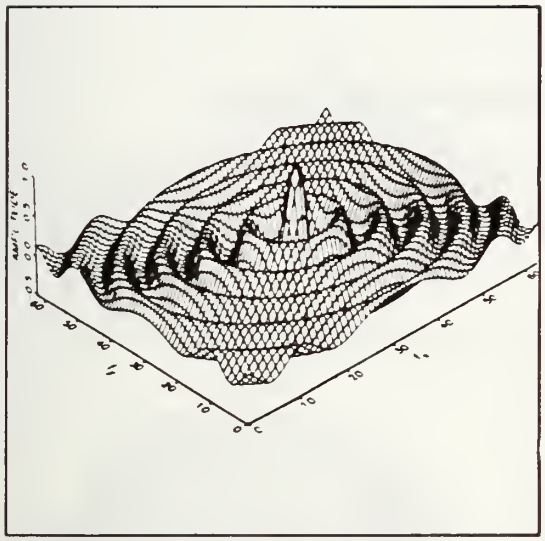

Figure 61. Time slice 23.

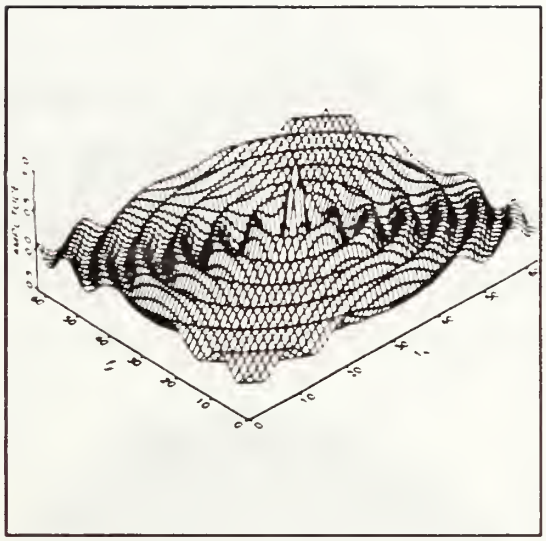

Figure 62. Time slice 28.

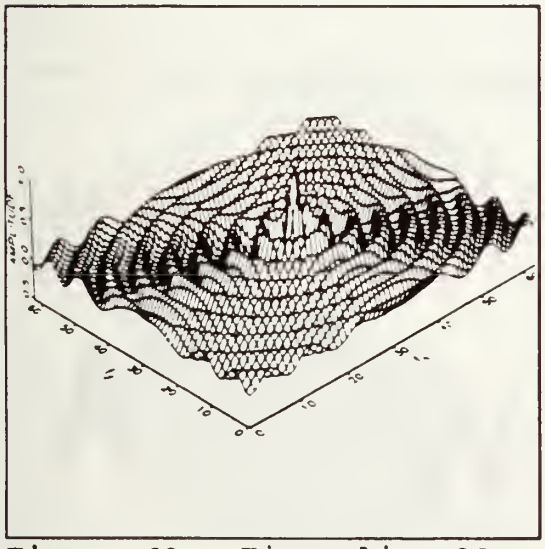

Figure 63. Time slice 38.

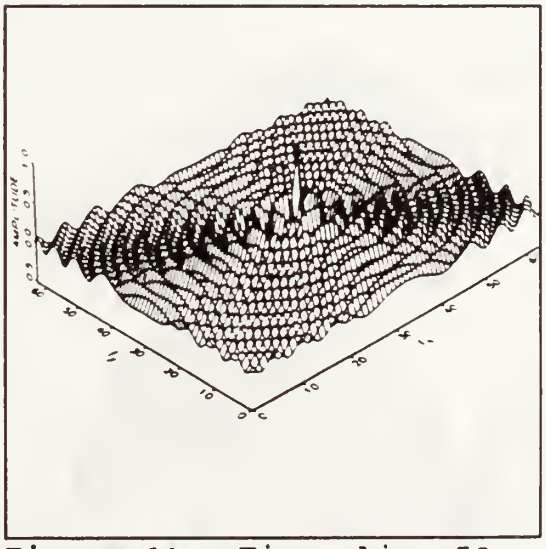

Figure 64. Time slice 58.

APPENDIX B. EXAMPLES OF DERIVATIVE FILTERS

This appendix contains additional examples of derivative filters. The value of time at which each was calculated in found in the figure caption.

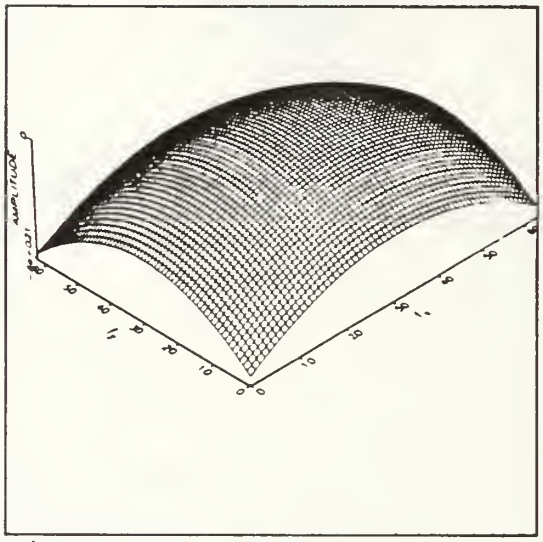

Figure 65. Time slice 1.

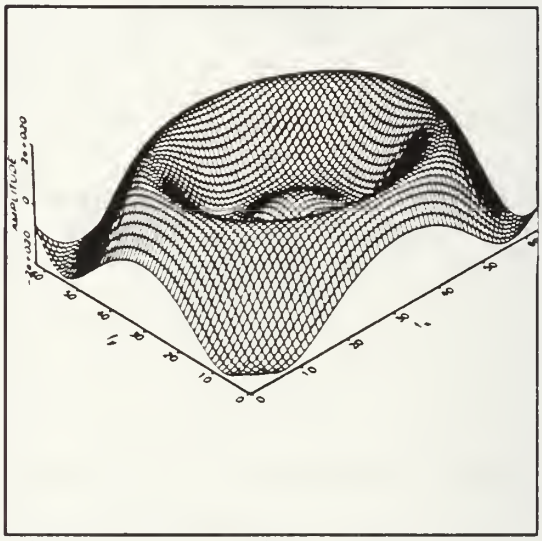

Figure 66. Time slice 3.

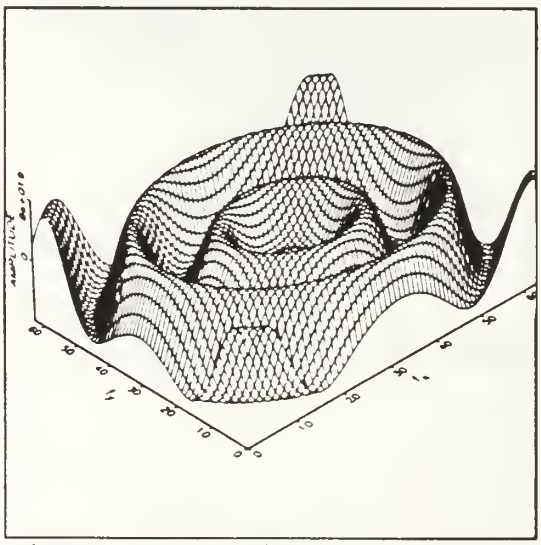

Figure 67. Time slice 8.

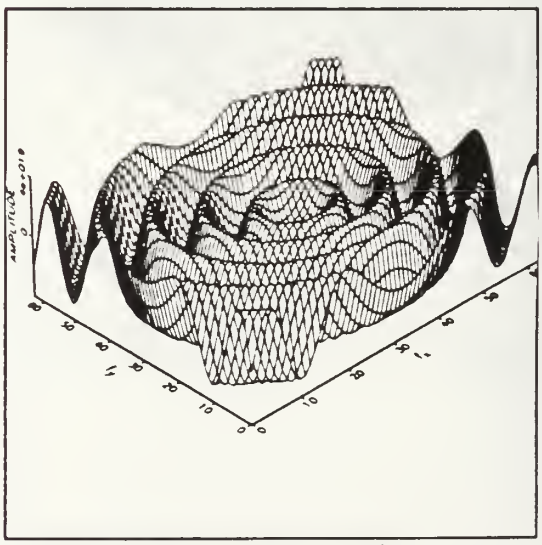

Figure 68. Time slice 18.

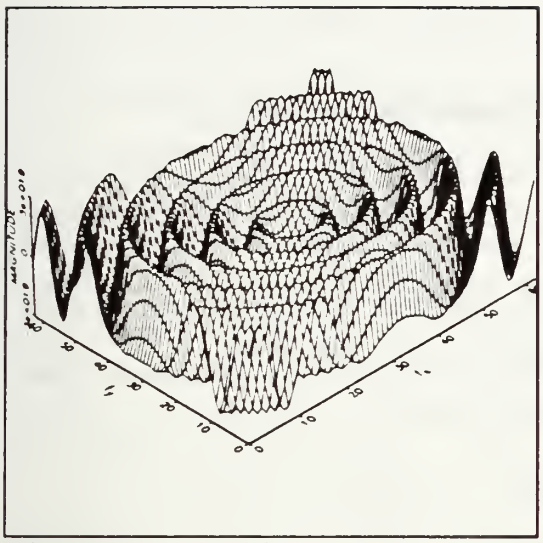

Figure 69. Time slice 23.

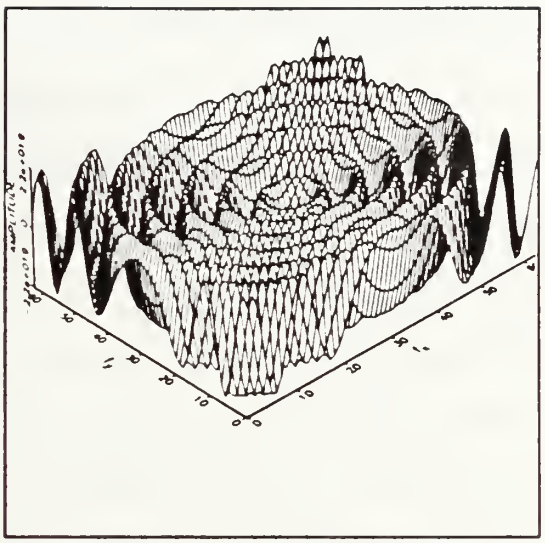

Figure 70. Time slice 28.

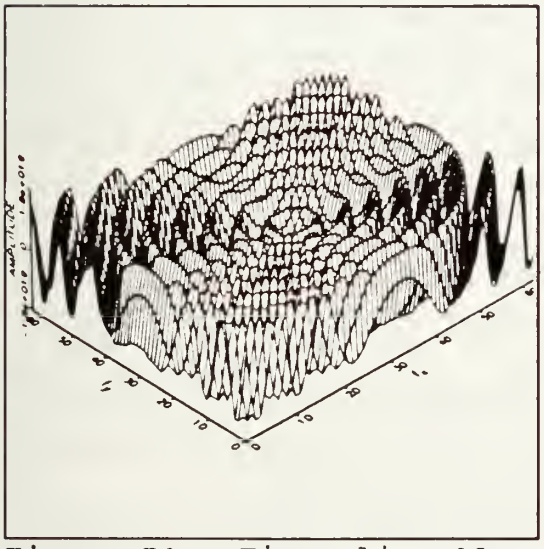

Figure 71. Time slice 38.

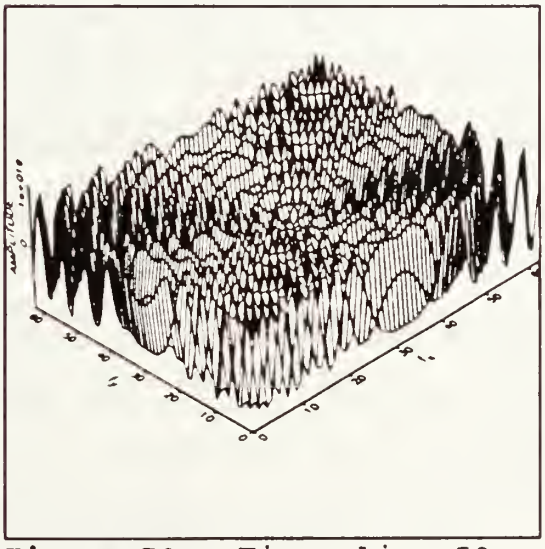

Figure 72. Time slice 58.

APPENDIX C. DETAILED EXPLANATION OF OPTFIL

This explanation addresses in detail the first module of the program--OPTFIL. Its primary function is to calculate the required Bessel filters for use by OPTPROP in completing the diffractive field calculations. This discussion assumes some knowledge of MATLAB programming and a familiarity with the information already presented. Refer to Appendix E for a copy of the code itself.

Initially, OPTFIL clears all variables in the work space and deletes any previously generated output files. It may be desirable to comment out this latter command if only the size of the transform space matrix is changing and not any of the defining parameters. In that manner, for example, 64x64 sized PROP matrices may be retained in their respective files while a 128x128 sized solution set is developed. Given sufficient space on the hard drive, two separate sets of data for a given combination of defining parameters are available to be used. Two sets will require the defining parameters to be saved in separate files. This is explained further below.

Next, the defining parameters are given specific values. This is not a user interactive procedure because of the time required to manually input nine variables per run when, in all likelihood, there will be very few changes once a set of

parameters has been decided upon. The definition of each is commented to the right of its respective variable.

Any matrix which will be generated during the run of the module is initialized by defining its size via the "zeros" command. The MATLAB User's Manual [Ref. 11] points out that this is a time saving step. All are square matrices except "rho_m" and "time" which have been designated vector matrices.

The "time" vector is next generated by the "linspace" command. It divides the time period from z/c to time_max into M-Step points. These are the values of time which will be used in the argument of the Bessel function.

Now that all of the necessary variables have been defined, including those needed by OPTPROP, the program saves them to disk by the "save" command. The name of the file immediately follows the command word and a space separates the file name from the variable to be stored within it. Note that this command allows multiple variables to be included within a single file. In this case, OPTBES is the file name and N, M, N0, Step, time, c, z, and t_eps are all stored. If a previously generated set of Bessel functions with a different size matrix is to be retained on the disk, a different file name will be required. This is necessary to provide OPTPROP with the proper value of N. OPTPROP will load the variables as OPTPROP begins to run.

Calculation of the spatial radius, rho, was somewhat tricky because of the varying size of the matrix quadrants.

The cartesian equivalents to radial values of rho are generated using the "meshdom" command. This command is oriented from the view of quadrant I. The shorter axis in the case of an NxN final matrix has N0-1 points (quadrant I is N0x(N0-1)). The command "linspace" was again used to divide the length of rho into N0-1 equidistant points to create the vector "rho_m". However, a length of N points exists in other quadrants which requires adding an additional increment of "rho_m". The vector "rho_m" has this additional increment added to it in the next line. "Rho_x" and "rho_y" are the cartesian equivalents of the radial vector "rho_m" and they are generated through the "meshdom" command. A matrix of radial distance values called "row" (to differentiate it from "rho") is created by applying the Pythagorean theorem to "rhox" and "rhoy". This is done once outside the upcoming loop rather than repeatedly within the loop.

The purpose of the loop is to sequence the Bessel argument through the values of time contained in the time vector. M-Step is the maximum number of time values and so the loop progresses from m=1 to m=M-Step. The "fprintf" command provides a screen display of the loop count for an indication of the progress through the loop. The first PROP matrices to be formed are those using the value of time within the time vector. Recall that these matrices are delineated through a suffix of "A". The full argument for the Bessel funcion is defined as "arg" and it is within "arg" that the value of time

progresses. A temporary matrix called "temp" holds the result of the Bessel operation on the "arg" matrix.

Taking advantage of the symmetry involved, the next sequence of instructions builds the full NxN matrix PROP from the N0xN0 matrix "temp". The first quadrant of PROP is formed initially. Since it has dimensions of N0x(N0-1), all of the row values of "temp" are used but only the first N0-1 columns. To form the second quadrant (N0xN0) requires the entire "temp" matrix. "Temp" must be flipped about the center to maintain the proper relationship of distance, and the command "fliplr" accomplishes this. Now that the entire top portion of the PROP matrix has been formed, the lower half is constructed by flipping the upper half about the center row. Because the bottom half has one less row than the top, only rows 2:N0 are flipped.

The next segment of the code correlates the PROP matrix to the time index. The variable "vname" is given the name "PROP(m)A". "Vname" is then set equal to the PROP matrix just formed. This result is stored to disk using the "save" command in a file named "p(N)x(m)A". The variables PROP and vname are cleared following this to save working space in RAM. Note use of the preparatory command "eval" in performing these last tasks.

This completes the code which generates the "A" PROP matrices; a similar sequence of commands, with one exception, forms the "B" PROP matrices. That exception is found within

the definition of "arg". The time value adds t_eps to that of the value within the time vector. Recall that these "B" PROP's are used to calculate the derivative filter via a difference equation approximation.

APPENDIX D. DETAILED EXPLANATION OF OPTPROP

This explanation addresses in detail the second module of the simulation--OPTPROP. Its primary function is to complete the field strength calculation once provided the appropriate Bessel filters generated by OPTFIL. This discussion assumes some knowledge of MATLAB programming and a familiarity with the information already presented. Refer to Appendix F for a copy of the code itself.

Initially, OPTPROP clears all variables in the work space and deletes any files no longer desired. Next, it loads from disk the defining parameters specified in OPTFIL and contained in the file "OPTBES" (if an additional file with different parameters has been created in OPTFIL, ensure the proper name is called by OPTPROP). This latter step is done very simply with the "load" command.

Following this, the excitation function is selected through interactive screen text. The value of N is first displayed as a reminder of the matrix size and the choice of the four available input functions is presented. The "disp" command is used repeatedly to provide the numerous lines of screen text. The desired excitation function (either circle, table, circular Gaussian or circular Bessel) is selected by keyboard entry of its respective excitation function number, as shown on the screen ("enter" must be struck after the

number is typed). A series of "if" and "elseif" logic statements follows which call the proper function "m" file. These function "m" files numerically model the excitation functions and copies of their code are found in Appendix G. All the function "m" files require either the diameter or the width to be specified as part of the call information. This information is again provided by the user interactively via the keyboard. A default value is shown on the screen in brackets. (Again "enter" must be keyed following the selection.) This value of diameter or width must be an odd number for the function "m" file to run properly and there is a reminder in the screen text to that effect. Input of an even value will result in an error message and the program OPTPROP will have to re-initiated. Other information may be also be required as appropriate to the particular function. For example, if the circular Gaussian excitation function is designated, the screen text will ask for the desired standard deviation, sigma.

The excitation function is pictorially displayed on the screen by the "mesh" command and is titled 'SHFT_INPUT'. It is then changed into the corner geometry in preparation for the two-dimensional Fourier transform operation. The "fftshift" command performs the first action and the "fft2" command accomplishes the second. Note that the result of any particular operation on the input function can be pictorially displayed by invoking the "mesh" command. As written in

Appendix F, the OPTPROP code has these subsequent "mesh" commands included, but commented out. The transformed excitation function, "F_input", is shifted back into the center of the transform space in preparation of multiplication with the Bessel filter (the appropriate PROP matrix). Now called "Fshft_input", it is equal to $\tilde{s}(x,y)$ in Equation 26.

The loop through the time vector commences at this point. It begins with m=1 and terminates at m=M-Step. The "fprintf" statement provides a screen display of the progress of the loop. The "A" file correlated to the time vector index "m" is "p(N)x(m)A". A new variable named "filename1" is set equal to the character string "p(N)x(m)A". File "p(N)x(m)B" is, in essence, renamed "filename2" in a similar manner. This renaming was done to simplify the upcoming commands. "Filename1" and "filename2" are loaded using the "load" command and a preparatory "eval" command. Each file contains its respective PROP matrix (PROP(m)A/B). New matrices, named "vname1" and "vname2", respectively, are created having the same values as PROP(m)A/B. This also was performed to simplify upcoming commands.

"Fshft_output1" is formed by the element by element product of "vname1" and "Fshft_input". It can be meshed and the title will reflect the loop sequence number (the same number as the time vector index). An example of the conversion from the .mat file to ASCII format is shown here. There will be other upcoming steps which have this code for

the conversion incorporated also, but it will not be pointed out any further. "Der_fil" is calculated by the difference equation approximation method discussed in Chapter 3. "Fshft_output2" is the result of the element-by-element product of "Der_fil" and "Fshft_input". At this point "vname"1, "vname2", "Der_fil", and the PROPA/B pair are cleared to free RAM.

To prepare for the two-dimensional inverse Fourier transform operation, both "Fshft_output1" and "Fshft_output2" are moved to the corner geometry, again by the "fftshift" command. This results in "F_output1" and "F_output2" being formed. Both the inverse transform operation and multiplication by the appropriate constants are done in the next step to produce "output1" and "output2". Each is shifted to the center geometry prior to the addition of the two being performed. The sum of "shft_output1" and "shft_output2" is called "shft_output". The "abs" command is applied to take the magnitude of "shft_output" and the result is designated "shft_outabs". Recall that the magnitude of "shft_output" is necessary to properly illustrate the output of a square law optical detector.

At this point the N0 or center row of "shft_outabs" is saved in the m+Step column of a new matrix called output_plot. MATLAB automatically puts zeros into the elements of the first "Step" number of columns. (Recall that this simulates the Heaviside step function.) The code provides the option to

similarly view the magnitude over time of "shft_output1" or "shft_output2" to facilitate the analysis of either. "Output_plot1" and "output_plot2" are the respective matrices formed. The end of the loop has now been reached.

When the loop has sequenced through all the values of time in the time vector and the matrix "output_plot" is complete, the data is stored to disk in a file named "o(d)x(M). The "o" signifies output, "d" is the diameter or the width of the excitation function, and "M" is the total number of time slices including Step. An example of this filename is "o17x64". Note that there is no indication of the specific type of excitation function used, only the diameter or width is specified in the filename. This file saved to disk allows access later to the completed output whether for view, plotting, etc.

Last comes a series of mesh plots to provide viewing "output_plot" from several aspect angles. The "subplot" command sets up two half-screen views rotated ninety degrees from each other. This display is kept on the screen for five seconds by the "pause" command before a full screen look is generated. This completes OPTPROP.

APPENDIX E. SOURCE CODE FOR OPTFIL

This appendix contains the source code for OPTFIL. The code is written using the MATLAB software package.

OPTFIL

```
%% OPTFIL.M
%% This program generates the Bessel filter.
%%   June 6,1992
clear;
%!del p*x*.mat
N = 64;                  % size of square array
M = 64;                  % number of time slices
N0 = (N/2)+1;            % defines center of the square array
Step = 3;                % number of leading zero time slices.
z = 80e-3;               % distance to the observation plane
time_max = .95e-9;       % time at the final time slice
rho = 200;                   % spatial radius of the filter
[sqrt(rhox^2
                         % + rhoy^2)]
c = 3e8;                 % velocity of the light wave
t_eps = 23.5e-25;       % time difference for the differential

%% Initialize matrices to save processing time
PROP = zeros(N);
temp = zeros(N0);
arg = zeros(N0);
rho_m = zeros(N0,1);
row = zeros(N0);
time = zeros(M-Step,1);

%% Generate M-Step time slices between z/c and time_max.
time = linspace(z/c,time_max,M-Step);

%% Save those variables necessary for OPTPROP.m in a file
%%called OPTBES.
save OPTBES N M N0 Step time c z t_eps;

%% Generate N0-1 values of rho_m from 0 to rho.
rho_m = linspace(0,rho,N0-1);

%% Add additional increment to rho_m to compensate for the
%%off-center orientation of the final NxN matrix
rho_m = [rho_m (rho_m(N0-1)+rho_m(2))];
```

```
%% Create two N0 x N0 arrays of rho values for function
%%evaluation.
[rhox,rhoy] = meshdom(rho_m,rho_m);

%% Calculate matrix of radial distance values outside the loop
row = sqrt(rhox.^2 + rhoy.^2);

for m = 1:M - Step          %%%%%%%%%%START LOOP%%%%%%%%%%
     fprintf('%3.0f',m);          %show m value on screen

   %% Generate PROP matrices with suffix of "A" corresponding
   %% to the values of the time vector
   %% Create an N0 x N0 array of argument values for the
   %% bessel function
     arg = row * sqrt( c^2*(time(m))^2-z^2 );

     % Evaluate J_0 at each argument value; creates an N0 x N0
     % array
     temp = besseln(0,arg);

     %  Create PROP matrix containing the Bessel filter data
     PROP(1:N0,N0:N) = temp(1:N0,1:N0-1);
     PROP(1:N0,1:N0) = fliplr(temp);
     PROP(N0:N,1:N) = flipud(PROP(2:N0,1:N));

     %Correlate the name of the variable PROP with the time
     %index;ie, PROP1A, PROP2A,...
     vname = ['PROP',int2str(m),'A'];       %set up name
     eval([vname,'= PROP ;']);

     %Save applicable PROP in a file named p(N)x(m)A;.e.g.,
     %PROP5A in p64x5A
     eval(['save p',int2str(N),'x',int2str(m),'A',' ',vname]);
     eval(['clear PROP ',vname]);

     %Generate PROP matrices with suffix of "B" corresponding
     %to the values of the time vector + t_eps

     % Create an N0 x N0 array of argument values for the
     % bessel function
     arg = row * sqrt( c^2*(time(m)+t_eps)^2-z^2 );

     % Evaluate J_0 at each argument value; creates an N0 x N0
     % array
     temp = besseln(0,arg);

     % Create PROP matrix containing Bessel filter data
     PROP(1:N0,N0:N) = temp(1:N0,1:N0-1);
     PROP(1:N0,1:N0) = fliplr(temp);
     PROP(N0:N,1:N) = flipud(PROP(2:N0,1:N));
```

```
      %Correlate the name of the variable PROP with the time
      %index;ie, PROP1B, PROP2B,...
      vname = ['PROP',int2str(m),'B'];       %set up name
      eval([vname,'=PROP ;']);

      %Save applicable PROP in a file named p(N)x(m)B;e.g.,
      %PROP6B in p64x6B
      eval(['save p',int2str(N),'x',int2str(m),'B',' ',vname]);
      eval(['clear PROP ',vname]);

end             %%%%%%%%%%END LOOP%%%%%%%%%%
```

APPENDIX F. SOURCE CODE FOR OPTPROP

This appendix contains the source code for OPTPROP. The code is written using the MATLAB software package.

OPTPROP

```
%% OPTPROP.m performs transient optical wave propagation
%% simulations. It uses the NxN arrays "p(N)x(m)A/B" to
%% compute the propagation transfer function.
%% June 2, 1992
%% Size of the variables
%% NxN -- input functions; M-Step -- time slices.
%% NxM--output_plot

clear;
!del opt*.met
%!del optder?.dat

%% Load the defining parameters specified in OPTFIL.m
load OPTBES.MAT

%% Generate the INPUT function; plot it.
 N
 disp('N is the width of the base for each function')
 disp('  ')
 disp('Please select the excitation function')
 disp('                    1 - Circle        ')
 disp('                    2 - Table         ')
 disp('                    3 - Circular Gaussian   ')
 disp('                    4 - Circular Bessel   ')
 disp('  ')
 disp(' Please strike "Enter" after selection.')
 disp(' ')
 disp(' Default values are in [ ].')

 input_func = input('Please enter an excitation function
                    number [1] ');
     if isempty(input_func)
           input_func = 1
     end

 if input_func == 1,
  d = input('Please enter ODD diameter, [13], d =  ');
  if isempty(d)
  d = 13
  end
```

```
 shft_input = crcle(d,N);

elseif input_func == 2,
 w = input('Please enter ODD width, [25], w =  ');
 if isempty(w)
 w = 25
 end
 shft_input = table(w,N);
 d = w;

elseif input_func == 3,
 sigma = input('Please enter the standard deviation, [12],
                sigma =  ');
 if isempty(sigma)
 sigma = 12
 end
 d = input('Please enter the ODD diameter, [25], d =  ');
 if isempty(d)
 d = 25
 end
 shft_input = crcgaus(sigma,d,N);

elseif input_func == 4,
 a = input('Please enter the width scaling factor,[1], a= ');
 if isempty(a)
 a = 1
 end
 d = input('Please enter the ODD diameter, [25], d =  ');
 if isempty(d)
 d = 25
 end
 shft_input = crcbess(a,d,N);

else
 error('Incorrect Excitation Function Selection')
end
mesh(shft_input);title('SHFT_INPUT');
   %following code used to enable print function and save
   %data to disk
      %meta optin
      %pause(1)
      %clg
      %vname = ['shft_input'];
      %eval(['save opshftin ',vname])
      %eval(['save opshftin.dat ',vname,' /ascii'])

%% Shift input quadrants and take the 2-D FFT to produce
%% F_INPUT.
input = (fftshift(shft_input));
   %following code used to observe and save data to disk
      %mesh(input);title('INPUT ')
```

```
       %pause(1);clg
       %vname = ['input'];
       %eval(['save optin ',vname])
       %eval(['save optin.dat ',vname,' /ascii'])
F_input = fft2(input);
       %mesh(F_input);title('F_INPUT ')
       %pause(1);clg

%% Shift F_input in preparation of multiplication with PROP
Fshft_input = fftshift(F_input);
       %mesh(Fshft_input);title('FSHFT_INPUT ')
       %pause(1);meta optf_in;clg

%% Plot the absolute value of the (shifted) transform for
%% viewing only
       %mesh(abs(Fshft_input)); title('ABS(FSHIFT_INPUT)')
       %pause(1);clg

%% Array-multiply the shifted transfer function PROP and
%% Fshft_input.
disp('Performing array multiplication....');
for m = 1:M-Step
 %%%%%%%%%%%% Start loop %%%%%%%%%%%%%%%%%%%%
    fprintf('%2.0f,',m)
                pause(1)
    filename1 = ['p',int2str(N),'x',int2str(m),'A' ];
    eval(['load ',filename1] );
    filename2 = ['p',int2str(N),'x',int2str(m),'B' ];
    eval(['load ',filename2] );
    eval(['vname1 = PROP',int2str(m),'A',';']);
    eval(['vname2 = PROP',int2str(m),'B',';']);
    Fshft_output1 = (vname1 .* (Fshft_input));
       %following code used to observe at given time slice (m)
       %and save data to disk
%           if m == 46
%                 mesh(Fshft_output1);
%                 title(['FSHFT_OUTPUT1 (',int2str(m),')'])
%                 pause(1)
%                 meta opt1;
%             vname = ['F_out1',int2str(m)];
%             eval([vname,'=Fshft_output1 ;'])
%             eval(['save optf1_',int2str(m),' ',vname])
%             eval(['save optf1_',int2str(m),'.dat ',vname,'
%                   /ascii'])
%           end
%           clg

    Der_fil = ((vname2/(time(m)+t_eps)) - (vname1/time(m)))...
                    /t_eps;
       %following code used to observe at given time slice (m)
       %and save data to disk
```

```
%           if m == 46
%                 mesh(Der_fil);
%                 title(['DER_FIL  (',int2str(m),')'])
%                 pause(1)
%                 meta optdir;
%             vname = ['Der_fil',int2str(m)];
%             eval([vname,'=Der_fil ;'])
%             eval(['save optdir',int2str(m),' ',vname])
%             eval(['save optdir',int2str(m),'.dat ',vname,'
%                 /ascii'])
%           end
%           clg

    Fshft_output2 = Der_fil .*(Fshft_input);
       %following code used to observe at given time slice (m)
       %and save data to disk
%           if m == 46
%                 mesh(Fshft_output2)
%                 title(['FSHFT_OUTPUT2 (',int2str(m),')'])
%                 pause(1)
%                 meta opt2
%             vname = ['F_out2',int2str(m)];
%             eval([vname,'=Fshft_output2 ;'])
%             eval(['save optf2_',int2str(m),' ',vname])
%             eval(['save optf2_',int2str(m),'.dat ',vname,'
%                 /ascii'])
%           end
%           clg

    %% Clear unnecessary variables to free RAM
       clear vname1; clear vname2; clear Der_fil;
       eval(['clear PROP',int2str(m),'A',';']);
       eval(['clear PROP',int2str(m),'B',';']);

    %% Shift Fshft_output(1,2) to corner geometry prior to
    %%taking the IFFT2
    F_output1 = fftshift(Fshft_output1);
%             mesh(F_output1) ; title('F_OUTPUT1 ')
%             pause(1);
%             clg;
    F_output2 = fftshift(Fshft_output2);
%             mesh(F_output2) ; title('F_OUTPUT2 ');
%             pause(1);
%             clg;

    %% Take IFFT of F_output(1,2) and multiply each by its
    %% respective constants to produce the outputs
    output1 = (ifft2(F_output1))*( 2*z/((c^2)*(time(m)^2)) );
%             mesh(output1); title('OUTPUT1 ');
%             pause(1);
%             clg;
```

```
    output2 =  (ifft2(F_output2))*(2*z/(c^2));
%               mesh(output2) ; title('OUTPUT2 ');
%               pause(1);
%               clg;

    %% Shift output(1,2) prior to summation
    shft_output1 = fftshift(output1);
       %following code used to observe at given time slice (m)
       %and save data to disk
%           if m == 46
%                 mesh(shft_output1) ;
%                 title(['SHFT_OUTPUT1 (',int2str(m),')'])
%                 pause(1);
%                 meta optout1
%             vname = ['out1',int2str(m)];
%             eval([vname,'=shft_output1 ;'])
%             eval(['save opt1_',int2str(m),' ',vname])
%             eval(['save opt1_',int2str(m),'.dat ',vname,'
                  /ascii'])
%           end
%           clg;
    shft_output2 = fftshift(output2);
       %following code used to observe at given time slice (m)
       %and save data to disk
%           if m== 46
%                 mesh(shft_output2) ;
%                 title(['SHFT_OUTPUT2 (',int2str(m),')'])
%                 pause(1)
%                 meta optout2
%              vname = ['out2',int2str(m)];
%              eval([vname,'=shft_output2 ;'])
%              eval(['save opt2_',int2str(m),' ',vname])
%              eval(['save opt2_',int2str(m),'.dat ',vname,'
                   /ascii'])
%           end
%           clg

    %% Calculate the shifted output
    shft_output = shft_output1 + shft_output2;
       %following code used to observe at given time slice (m)
       %and save data to disk
%           if m == 46
%                 mesh(shft_output);
%                 title(['SHFT_OUTPUT (',int2str(m),')'])
%                 pause(1)
%                 meta optout
%              vname = ['out',int2str(m)];
%              eval([vname,'=shft_output ;'])
%              eval(['save opt3_',int2str(m),' ',vname])
%              eval(['save opt3_',int2str(m),'.dat ',vname,'
                   /ascii'])
```

```
%           end
%           clg

    %% View the magnitude of the shifted output
       shft_outabs = abs(shft_output);
       %following code used to observe at given time slice (m)
       %and save data to disk
%           if m == 46
%                 mesh(shft_output);
%               title(['MAG SHFT_OUTPUT (',int2str(m),')'])
%                 pause(1)
%                 meta optabs
%              vname = ['outabs',int2str(m)];
%              eval([vname,'=shft_outabs ;'])
%              eval(['save optab_',int2str(m),' ',vname])
%            eval(['save optab_',int2str(m),'.dat ',vname,'
                  /ascii'])
%           end
%           clg

    %% Save the N0 row of the magnitude of the shifted output1
    %% in the m/2 column of output_plot1.
    output_plot1(1:N,m+Step) = abs(shft_output1(N0,1:N))';

    %% Save the N0 row of the magnitude of the shifted output2
    %% in the m/2+Step column of output_plot2.
    output_plot2(1:N,m+Step) = abs(shft_output2(N0,1:N))';

    %% Save the N0 row of the magnitude of the shifted output
    %% in the m+Step column of output_plot.
    output_plot(1:N,m+Step) = shft_outabs(N0,1:N)';

end
%%%%%%%%% End loop %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%       Save contents of "output_plot" as NxM array.
filename = ['o',int2str(d),'x',int2str(M)];  % File: o(d)x(M)
eval(['save ',filename,' output_plot'] );

%% Plot the absolute value of "output_plot".
subplot(121), mesh(rot90(output_plot,1));
subplot(122), mesh(rot90(output_plot,2));
pause(5)
%% Plot full screen view
clg;
mesh(rot90(output_plot,3));
```

APPENDIX G. SOURCE CODE FOR EXCITATION FUNCTIONS

This appendix contains the source code for each of the four excitation functions. These functions are the uniform circular and uniform square functions, and the circular Gaussian and circular Bessel functions.

UNIFORM CIRCULAR EXCITATION FUNCTION

```
function Y = crcle(d,N)
%     CRCLE.M:  Y = crcle(d,N)
% Program for generating uniform circular excitation functions
%    d is the DIAMETER of the circle.  (ODD integer)
%    N is the WIDTH of the square base. (EVEN integer)
%    Example: z = crcle(33,64);
%%%%%        JG Upton     March, 1992

%    Check that d is an odd integer
     if rem(d,2) < 0.1;
     error('The diameter of the crcle function must be an ODD
integer.');
     else;
     end;

%    Check that N is an even integer
     if rem(N,2) ~= 0.0;
     error('The width of the square base must be an EVEN
integer.');
     else;
     end;

N0 = (N/2)+1;                          %N0 is the center location
r = d/2;                                %r is the radius
Y = zeros(N);
temp = zeros(N0-1);

for m = 1:r+1;
    for n = 1:r+1;
        if sqrt((m-1)^2 + (n-1)^2) <= r;
        temp(m,n) = 1;
        end;
    end;
end;
```

```
Y(N0:N,N0:N) = temp;
Y(2:N0,N0:N) = flipud(temp);
Y(2:N0,2:N0) = rot90(temp,2);
Y(N0:N,2:N0) = fliplr(temp);
%%%%%For testing purposes:  mesh(Y)
```

UNIFORM SQUARE EXCITATION FUNCTION

```
function Y = table(w,N)
%     TABLE.M: Y = table(w,N)
%Program for generating a uniform square excitation function.
%   w is the WIDTH of the table.  (ODD integer)
%   N is the WIDTH of the square base.  (EVEN integer)
%   Example:  z = table(33,64);
%%%%%       JG Upton    March, 1992.

%     Check that w is an odd integer.
      if rem(w,2) < 0.1;
      error('The width of the table must be an ODD integer.');
      else;
      end;

%     Check that N is an even integer
      if rem(N,2) ~= 0.0;
      error('The width of the square base must be an EVEN
integer.');
      else;
      end;

N0 = (N/2)+1;                                  % N0 is the center
location
w0 = ceil(w/2);                              % w0 is the mid-point of
the table
Y = zeros(N);
temp = zeros(N0-1);
temp(1:w0,1:w0) = ones(w0);

Y(N0:N,N0:N) = temp;
Y(2:N0,N0:N) = rot90(temp);
Y(2:N0,2:N0) = rot90(temp,2);
Y(N0:N,2:N0) = rot90(temp,3);
%%%%%For testing purposes:  mesh(Y)
```

CIRCULAR GAUSSIAN EXCITATION FUNCTION

```
function Y = crcgaus(sigma,d,N)
%     CRCGAUS.M:  Y = crcgaus(sigma,d,N)
```

```
%Program for generating circular Gaussian excitation
functions.
%   sigma is the STANDARD DEVIATION of the gaussian function.
%   d is the  DIAMETER of circle.  (ODD integer)
%   N is the WIDTH of the square base. (EVEN integer)
%   Example: z = crcgaus(12,33,64);
%%%%%        JG Upton   March, 1992.

mu=0;                                   %mu is the mean of the
gaussian function

%     Check that d is an odd integer.
      if rem(d,2) < 0.1;
      error('The diameter of the circle function must be an
ODD integer.');
      else;
      end;

%     Check that N is an even integer.
      if rem(N,2) ~= 0.0;
      error('The width of the square base must be an EVEN
integer.');
      else;
      end;

N0 = (N/2)+1;                       % N0 is center location;
r = d/2;                            % r is the radius
Y = zeros(N);
temp = zeros(N0-1);

for m = 1:(d+1)/2;
    for n = 1:(d+1)/2;
        x = sqrt((m-1)^2+(n-1)^2);
        if x <= r;
        temp(m,n) = (1/(sqrt(2*pi)*sigma))*exp(-((x-mu)^2)/...
                    (2*(sigma^2)));
        end;
    end;
end;

Y(N0:N,N0:N) = temp;
Y(2:N0,N0:N) = flipud(temp);
Y(2:N0,2:N0) = rot90(temp,2);
Y(N0:N,2:N0) = fliplr(temp);
%%%%%For testing purposes:  mesh(Y)
```

```
function Y = crcgaus(sigma,d,N)
%     CRCGAUS.M:  Y = crcgaus(sigma,d,N)
%Program for generating circular Gaussian excitation
functions.
%   sigma is the STANDARD DEVIATION of the gaussian function.
%   d is the  DIAMETER of circle.  (ODD integer)
%   N is the WIDTH of the square base. (EVEN integer)
%   Example: z = crcgaus(12,33,64);
%%%%%        JG Upton   March, 1992.

mu=0;                                    %mu is the mean of the
gaussian function

%     Check that d is an odd integer.
      if rem(d,2) < 0.1;
      error('The diameter of the circle function must be an
ODD integer.');
      else;
      end;

%     Check that N is an even integer.
      if rem(N,2) ~= 0.0;
      error('The width of the square base must be an EVEN
integer.');
      else;
      end;

N0 = (N/2)+1;                          % N0 is center location;
r = d/2;                               % r is the radius
Y = zeros(N);
temp = zeros(N0-1);

for m = 1:(d+1)/2;
    for n = 1:(d+1)/2;
        x = sqrt((m-1)^2+(n-1)^2);
        if x <= r;
        temp(m,n) = (1/(sqrt(2*pi)*sigma))*exp(-((x-mu)^2)/...
                    (2*(sigma^2)));
        end;
    end;
end;

Y(N0:N,N0:N) = temp;
Y(2:N0,N0:N) = flipud(temp);
Y(2:N0,2:N0) = rot90(temp,2);
Y(N0:N,2:N0) = fliplr(temp);
%%%%%For testing purposes:  mesh(Y)
```

CIRCULAR BESSEL EXCITATION FUNCTION

```
function Y = crcbess(a,d,N)
%     CRCBESS.M:  Y = crcbess(a,d,N)
% Program for generating circular Bessel excitation functions.
%    a is the WIDTH SCALING FACTOR.
%    d is the DIAMETER of the circle.  (ODD integer)
%    N is the WIDTH of the square base. (EVEN integer)
%    Example: z = crcbess(1,33,64);
%%%%          J.G. Upton   March, 1992

%       Check that d is an odd integer.
        if rem(d,2) < 0.1;
        error('The diameter of the circle must be an ODD
integer');
        else;
        end;

%       Check that N is an even integer.
        if rem(N,2) ~= 0.0;
        error('The width of the square base must be an EVEN
integer');
        else;
        end;

N0 = (N/2)+1;                                %N0 is the center
location
r = d/2;                                  %r is the radius of the
circle
Y = zeros(N);
temp = zeros(N0-1);

for m = 1:r+1;
    for n = 1:r+1;
        x = sqrt((m-1)^2 + (n-1)^2);
        if x <= r;
        temp(m,n)=besseln(0,a*x);
        end;
     end;
 end;

%
Y(N0:N,N0:N) = temp;
Y(2:N0,N0:N) = flipud(temp);
Y(2:N0,2:N0) = rot90(temp,2);
Y(N0:N,2:N0) = fliplr(temp);

%%%%%For testing purposes:  mesh(Y)
```

LIST OF REFERENCES

1. J. W. Goodman, Introduction to Fourier Optics. San Francisco, CA: McGraw-Hill, Inc., 1968.

2. D. Guyomar and J. Powers, "Propagation of transient acoustic waves in lossy and lossless media," in Acoustical Imaging Volume 14, A. J. Berkhout, J. Rider, and L. F. van der'Wal, Eds. New York, NY: Plenum Press, 1985, pp. 521-531.

3. D. Guyomar and J. Powers, "A Fourier Approach to Diffraction of Pulsed Ultrasonic Waves in Lossless Media", in Journal of Acoustical Society of America, vol. 82 no.1, pp. 354-359 (1987).

4. J. Powers and D. Guyomar, "Propagation of Transient Scalar Waves: A Fourier Optics Approach, In preparation.

5. T. Merrill, "A Transfer Function Approach to Scalar Wave Propagation in Lossy and Lossless Media", Master's Degree Thesis, Naval Postgraduate School, March 1987.

6. P. R. Stephanishen, "Transient radiation from pistons in an infinite planar baffle", in Journal of Acoustical Society of America, vol. 49, pp. 1629-1637 (1971).

7. P. R. Stephanishen, "Acoustic transients in the far-field of a baffled circular piston using the impulse response approach", J. Sound Vib., vol. 32, pp. 295-310 (1974).

8. P. R. Stephanishen, "Acoustic transients from planar axisymmetric vibrators using the impulse response method", Journal of Acoustical Society of America, vol. 70, pp. 1176-1181 (1981).

9. P. R. Stephanishen and G. Fisher, "Experimental verification of the impulse response method to evaluate transient acoustic fields", Journal of Acoustical Society of America, vol. 69, pp. 1610-1617 (1981).

10. G. R. Harris, "Review of transient field theory for a baffled planar piston", Journal of Acoustical Society of America, vol. 70, pp. 10-20 (1981).

11. MATLAB for MS-DOS Personal Computers, User's Guide by The MathWorks, Inc., Natick, ME (1990).

12. AXUM, Technical Graphics and Data Analysis, User's Manual by TriMetrix, Inc., Seattle, WA (1989).

BIBLIOGRAPHY

1. R. M. Bracewell, The Fourier Transform and Its Applications. San Francisco, CA: McGraw-Hill, Inc., 1965.

2. E. Oran Brigham, The Fast Fourier Transform and Its Applicatons. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1988.

3. Partha P. Banerjee and Ting_Chung Poon, Principles of Applied Optics. Boston, MA: Irwin, Inc., and Aksen Associates, Inc., 1991.

4. A. V. Oppenheim and A. S. Willsky with I. T. Young, Signals and Systems. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1983.

5. C. A. Balanis, Advanced Engineering Electromagnetics. New York, NY: John Wiley and Sons, 1989.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	1
4.	Professor John P. Powers, Code EC/Po Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	4
5.	Professor Ron J. Pieper, Code EC/Pr Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	1
6.	Commandant of the Marine Corps Code TE 06 Headquarters, U. S. Marine Corps Washington, D.C. 20380-0001	1
7.	LtCol. John G. Upton USMC One Surf Way #123 Monterey, California 93940	2

Thesis
U4584
c.1

Upton
Microcomputers simulation of a Fourier approach to optical wave propagation.